JPRS-UCC-84-004

11 JULY 1984

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

FBIS FOREIGN BROADCAST INFORMATION SERVICE

JPRS-UCC-84-004

11 July 1984

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

CONTENTS

PERSONALITIES

GENERAL

ORGANIZATIONAL PROBLEMS IN DEVELOPMENT, REPAIR OF COMPUTERS

Moscow TRUD in Russian 20 Jan 84 p 2

[Article by Academician Yu. Nesterikhin, director of the Institute of Automation and Electrometry, USSR Academy of Sciences, Siberian Branch: "Computers: Possibilities and Problems"]

[Text] Novosibirsk--When I hear people saying that our country needs hundreds of thousands of computers or hear them citing huge numbers of operational computers in telling how rapidly we are approaching this goal, I am not quick to rejoice. The fact is that, at the present level of dependability, more than 15 people are needed to service an average machine. Hundreds of thousands of on-line computers engage millions of workers who are drawn away from production.

We are accustomed to the idea that an operator should be next to a machine--a youthful creature, proud in the knowledge of which buttons to press to start the machine. But an army of operators is merely a feeble and hopeless attempt to compensate for the shortcomings of the technology. It is becoming customary to hide the computer's downtimes and to cover up the defects that were allowed to enter into its design and manufacture. Between the level of computer equipment and that attained by industrial technology, there is a direct feedback relationship. Without modern technology, one cannot provide for the production of reliable and effective computer equipment, and without the contribution of the computer, it is impossible to reach the necessary level of modern technology. It would seem that today everyone should understand this simple truth. But organizational muddle and bureaucratic hurdles prevent the matter from receiving the attention it deserves.

Every department is interested in having modern technology. Virtually no department would be opposed to using computers. But from whom can they demand quality computers? With whom can they even reach an agreement? When many departments are responsible for the production of computer equipment, there is in effect no one responsible. This is why each ministry that has the most minimal facilities starts production of half-baked computers or computers that are borrowed in passing from somewhere. Should we be surprised at their poor quality?

Allow me to make one more point concerning the danger of departmentalization in general. Let us assume that at some stage in the development of computer equipment we should decide to allocate the production of different types of computers among departments on the basis of their industrial facilities... One department is responsible for producing large universal computers, another for manufacturing small ones, a third for making control computers, and the fourth for turning out computer components of all types. At first, it would seem to suit everyone, for it would spare manufacturers unnecessary competition. But time marches on, and the advance of technology and know-how has made it possible to produce a computer in a single semiconductor chip. And now it turns out that the department responsible for issuing components must itself produce computers. The bus principle is used, and thus a very powerful computer is constructed from units. To which department should such a computer be assigned? Because of its size, it should be assigned to the department that makes the large machines, but because of its component parts, probably, it should go to the department that handles the production of microcomputers...

It might seem that the solution lies in a periodic review of the dividing lines separating the departments. But this isn't right either. A very promising development is the creation of optical-electronic computers where part of the data is stored and processed in the form of light signals. There are already certain achievements in this field. The stumbling block, as usual, turns out to be a new technology--this time, the technology of producing the optical elements. Who will take up its development? Enterprises of the optics industry are under one ministry, but the class of computers that would include the optical-electronic ones is assigned to a different ministry. At this point, it is time to set up a new department. But organizational changes are hardly capable of accelerating the progress of technical innovations in production.

The best thing to hope for is a technical servicing system for computers. The speed of debugging the hardware is one of the most vulnerable points. It has almost become the rule to wait 3 to 4 months for the debugging of a machine. This applies to a computer that is produced in a few weeks. Outwardly, everything would seem to be satisfactory: the machine is built, its computing power is great, space is provided at the enterprise, and service personnel is hired according to the state norms. But debugging drags on. It is performed by an outside organization, and the longer it takes, the more money it makes.

The Soyuz EVM komplekt is an association set up for centralized servicing of computers. Similar firms in both capitalist and socialist countries are fully liable for each minute the computer is idle through their fault. In our country, the opposite is true. The organization which should bear the responsibility for the quality of work and the downtime stands between the consumer and the manufacturer because it has received the exclusive right to handle all sapre parts needed to service computers. Thus, even when a firm has the facilities to debug the machine, it is forced to turn to a service firm for spare parts.

The working method of service organizations whose representatives are called out by telephone is borrowed from the practice of such firms in the small, industrially developed countries. Where distances are insignificant and communications function superbly, such a system is justified. But in our country? The Novosibirsk department of the Soyuz EVM komplekt alone services a territory that exceeds that of non-Russian Europe. With distances like these, it is senseless to run computer repair services out of one center.

It is not merely a matter of the difficulty of forming new organizations. In order to make this clear, I will cite an example not directly connected with computer service. A series of scientific experiments required the simultaneous use of two computers, one in Moscow and the other in Novosibirsk. The USSR Ministry of Communications asked an enormous sum for this, charging a rate of 15 kopeks per minute: what you would pay for a private inter-city telephone conversation lasting 24 hours a day for an entire year. With a charge like this, the cost of a communications line for the two machines would equal the cost of one computer. We offered to pay not for a year of telephone time but for just the actual communications sessions. But the ministry did not agree. The reasoning is completely clear: on a session price schedule, additional trouble might occur if the connections could not be made for some reason. It is much simpler to sell the communication line outright for one solid year, thus avoiding unwanted bother and responsibility. Indeed, department interests badly serve the public cause.

Where is a way out? The existing situation can be changed by only one course of action: the creation of a union-wide goal-directed program and a powerful special subdivision like those organized for our country for working out technical ideas regarded by the government as especially vital and unifying the production output of the various departments. The experience of creating the best domestic computers shows that high results can be achieved primarily by forming a specially-tasked scientific production association with the help of a collective whose existence is dedicated exclusively to this specific task. If this work is passed from hand to hand, from scientific-research institute to design-engineering bureau, and from here to a pilot factory and from a pilot factory to serial production, there is no reason to trust in the effectiveness of the equipment developed. After all, each participant involved in making a new product has its own objectives and assignments, in no way related to the effectiveness of the new equipment.

It's precisely within the framework of a goal-directed program that union-wide measures could be provided to increase the reliability of computer equipment, to reduce labor costs for its operation and to set up a reliable system of servicing computers. It is now time to understand that the effective application of computers is increasingly becoming one of the essential ingredients in technical progress and a catalyst in the development of the country's economy.

9992
CSO: 1863/121

PROBLEMS IN STATUS AND TRAINING OF COMPUTER PROGRAMMERS

Moscow IZVESTIYA in Russian 3 Feb 84 p 2

[Article by A. Yershov, corresponding member, USSR Academy of Sciences: "Man and the Computer"]

[Text] Of all the technical achievements of mankind, none has developed so swiftly as that of computer technology. The total capacity of computers has been increasing by an order of magnitude every decade, but the cost of data processing has dropped many times over. And this answers society's growing demand for accuracy and speed in processing information.

Moving the economy onto the trail of intensive development, an increase in management efficiency, growth of labor productivity, adoption of resource-saving technology--all this requires information and knowledge that is up-to-date, complete, locally appropriate and prompt.

Whereas computer applications have previously been characteristic for science, new technology and the higher level of administrative, with the advent of microprocessors, computer technology will penetrate increasingly into each work place. Although it will occupy little space it will be the most vital component--the brain of each machine in industry and daily life.

The increasing dependence of society on computer technology also creates new problems. One of these is the problem of writing programs for computers.

A knowledge of language and mathematics and the ability to reason allow one to represent any mental task as a function of processing textual, numerical or symbolic information. The computer is a device capable of processing any kind of information at tremendous speeds, performing millions of operations per second.

But a computer can perform useful work only when for each operation performed by it there is a written program, a precise and detailed description of all operational sequences performed by the machine. Computer programming is becoming a wide-spread profession and at the same time one of the most difficult intellectual professions. A full-fledged industrial robot requires programs consisting of tens and even hundreds of thousands of commands. An on-board control system for a modern passenger plane requires even more: more than a million commands.

The difficulty of the programmer's profession is connected with the contradictory requirements placed on it. Being a creative activity, programming also demands scrupulous attention to purely formal details. In producing programs, one must be able to deal with both the abstractness of mathematical reasoning and the concreteness of engineering design. Assuming responsibility for the performance of a machine (an error could cost the economy millions of rubles), one must be able to rely on the work of others in creating program complexes. Lastly, programming has high requirements for attentiveness, precision, memory, quickness of apprehension and patience.

However, programming does not lay claim to any non-traditional abilities. The ability to plan one's actions, foresee their consequences and answer for them has always characterized intelligent, active and respectable persons. The problem lies in the fact that computers demand the development of these qualities to a new, high degree. It is important that the skills and knowledge of man be fully expressed in programs for robots and computers.

The first step on the road to this goal would be to grant computer programming the necessary professional status. It must be recognized as an independent profession with its own body of knowledge and skills, its own value system, its own professional and ethical code, comparable to the ethics of a physician or teacher. The profession's self-determination is objectively based on the appearance in the economy of a novel product: programs created for multiple and prolonged use in a large number of computers. Lastly, programs can become objects of personal use, like books and tape recordings.

The need to define the high social value and cost of the programmer's labor and to establish an economics of programming will serve as the economic foundation of professional programming. Economic incentives must be provided for the creation of programs that are highly reliable.

Reliability is presently becoming the number-one problem. Not many years ago, the work of the computer was closely supervised by people such as operators, engineers and programmers, who analyzed the performance of a machine and functioned as constant mediators between the machine and the outside world. Today the application of computer technology is assuming a new character. Almost without any human intervention, the computer is beginning to manage economic and technical entities and technology. These days, the computer not only controls landings of large airliners but likewise manages industrial processes where an error could entail catastrophic consequences.

For this reason, experts have developed principles of systematic and effective design for highly reliable programs. But there is still a long way to go before these achievements of science become general property. It is here that education faces its most vital tasks.

Programmers with a higher education are being prepared today chiefly within university speciality 0647: applied mathematics. Such preparation played a positive role in the 1970s, but programs of the traditional academic

curricula are increasingly falling short of present requirements. To express it more simply, the universities do not adequately teach programming, and the VUZes, applied mathematics.

There is, moreover, the fact that training in applied mathematics has a somewhat different methodological purpose, which is directed primarily to computer-assisted problem solving. In such an approach, a specialist is disposed to regard programming as an intermediate step in his work. Once the program is solved, the "user" immediately forgets the program he just created. For the professional programmer, the completion of a program is the end result of his productive efforts and the starting point for assuming full responsibility for the functioning of a computer that is equipped with his product.

Even greater challenges confront secondary education. These have been formulated in the CPSU CC project "Basic Directions for the Reform of General Educational and Professional Schools." One of the most important tasks in raising the quality of the teaching-educational process is identified here as follows: "To equip students with knowledge and skills for using modern computer technology, to provide widespread application of computers in the learning process and to establish special school and interschool study areas for this purpose."

Doing this will require large amounts of scientific research and a broad pedagogical experiment.

From the pool of secondary school graduates, a strong cadre of young people must be formed which will dedicate itself to the profession of programming.

Substantial changes must be made in the area of secondary vocational education. The spread of flexible automated production and the rise in the number of root-manipulators and automated work stations demand of the worker new production skills and a different level of culture. A significant portion of the graduates of polytechnical institutes must possess a clear understanding of the principles of computer equipment and its programming, and must be able to work with computer techniques and computer-based automation as operator-installers and operator-programmers.

In the SO [Siberian Branch] of the USSR Academy of Sciences, research and experiments have been underway for 10 years on the application of the computer to classroom teaching. Students were permitted access to the most modern equipment of the Computer Center and of other institutes of Akademgorodok. Research has shown that students have the greatest success in mastering computer technology when they are given the opportunity to use computers in their daily activities. During this academic year, preparation is underway for regular experimental lessons in the computer classroom for the fourth and seventh grades.

Disputes often occur among teachers concerning the degree to which the application of technical means of instruction can help solve the more fundamental tasks of education, in particular, the task of instilling

morality and communist ethics. It is important to note here that such a course of study for programming--seemingly so specific at first glance--actually embodies serious moral and ethical questions.

Computer networks, data systems, data banks and control systems--all of this together represents what might be called the central nervous system of society and contains enormous quantities of information about society. Programmers are becoming trusted custodians of privileged knowledge about information and the ways it is used. This is why programmers at all levels are faced with a most serious ethical problems: not to abuse this professional knowledge.

We must take care that, strengthening the link between knowledge and skill, ability and ethics, and personal enterprise and morality, we raise members of the new programming profession that are worthy of the age of developed socialist society.

9992
CSO: 1863/121

COMPUTER COLLECTIVE PLEDGES QUALITY FOR 1984

Minsk SOVETSKAYA BELORUSSIYA in Russian 8 Jan 84 p 1

[Article: "Excellent Quality for Electronics: the Socialist Pledges of the Collective of the Minsk Order of the October Revolution Production Association for Computer Equipment for 1984"]

[Text] Carrying out the historic decisions of the 26th Party Congress, the workers, technicians and employees of the association successfully fulfilled the tasks of the first three years of the Five-Year plan, satisfying all technical and economic indicators. Production volume increased by more than 42 percent. In the past year, serial production of one of the country's most powerful computers, the EC-1061, began. Pilot models of state-of-the-art computers were developed. Production sales exceeded the plan by the tens of millions of rubles.

Responding dutifully to the resolutions of the December (1983) Plenum of the CPSU Central Committee and those of the 9th session of the USSR Supreme Soviet, preparing for the worthy celebration of the 40th anniversary of the liberation of Belorussia from the German fascist aggressors and realizing the initiative "High Quality in the Development and Production of all Goods," the collective of the creators of Minsk computers have accepted the following socialist pledges.

By more fully utilizing internal resources and increasing organization and discipline in all units of production, to produce goods whose value is 11 million rubles beyond the plan, to guarantee the prompt fulfillment of contracted pledges for production delivery, to increase the production of goods for economic and domestic use by 14 percent.

To continue working toward a higher technical standard and higher quality in produced goods. To begin production of the new ES-1036 and ES-1065 computers. To reach a quality level where at least 68 percent of the goods undergoing certification bear the Stamp of Quality.

To carry out a set of measures designed to improve the method and technology of production, to introduce into operation 11 mechanized and automated sections and lines, 22 industrial robots and manipulators and to introduce 16 new industrial processes.

To achieve an increase in labor productivity of 1.5 percent in excess of the plan. To attain the entire increase in production volume by means of increased labor productivity.

To improve equipment utilization and raise the output-capital ratio by 3.5 percent.

By rational use of resources, to save 17 metric tons of nonferrous metals, 110 tons of ferrous metals and 145,000 kilowatt hours of electricity beyond established norms. By putting into use inventions and innovative suggestions, to achieve an annual savings of 2 million rubles. To reduce the labor intensity of production by 17.5 percent and to lower production costs by 0.5 percent in excess of the plan. To incorporate no less than 77 percent of the workers into the brigade form of organization and work incentives. To assimilate 52.5 percent of the technical workers into the active work of the creative collectives.

Using the association's capital, to construct 3,500 square meters of living space. To render assistance to the supporting kolkhozes and sovkhozes and to invest 100,000 rubles in the construction of buildings for private agricultural operations.

The socialist pledges were discussed and adopted at meetings of the association's labor collectives.

9992
CSO: 1863/121

UDC 681.326.7

HARDWARE-SOFTWARE FACILITIES IN DIAGNOSTIC SYSTEM FOR PS-2000 MULTIPROCESSOR

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
(manuscript received 9 Dec 81, after revision 23 Mar 83) pp 116-117

[Article by Nadezhda Vasil'yevna Belil'shchikova, senior scientific associate; Viktor Vasil'yevich Zastela, engineer; Il'ya Izrailevich Itenberg, candidate of engineering sciences; and Grigoriy Yur'yevich Pivovarov, engineer; all from NII UVM [Scientific Research Institute of Control Computers] (Severodonetsk)]

[Text] The large amount of equipment making up the PS-2000 multiprocessor imposes increased requirements for its diagnostic facilities.

To minimize mean time to recovery and increase the multiprocessor usage factor, hardware and software facilities for localizing malfunctions have been incorporated into the diagnostic system. The PS-2000 multiprocessor diagnostic facilities are: hardware facilities for on-line checking, software facilities for functional testing, and hardware-software facilities for localizing malfunctions.

The hardware facilities for on-line checking enable finding malfunctions in the memory, cable and memory refresh assemblies during multiprocessor operation.

Hamming code which allows finding and correcting single memory errors is used to check memory. When any error occurs (correctable or uncorrectable), an error signal is generated and the memory assembly is identified.

The error signal causes an interrupt in the monitor subsystem; the identifier of the memory assembly in which an error occurred is placed in the interrupt word register and can be polled by the monitor subsystem to identify the malfunction. The method of generating an interrupt for a single or double memory error is determined by interface commands sent from the monitor subsystem. Generation of an interrupt for a single memory error is used primarily in the mode of functional testing of memory assemblies.

Correctness of transmission of information over the cables is checked by parity. When an error in transmission to the monitor subsystem is detected, an interrupt signal is generated; the identifer of the malfunctioning cable is placed in the interrupt word register.

Memory refresh is checked by interval between adjacent refresh operations. If the maximum of 2 ms is exceeded by 0.1 ms, information in the multiprocessor memory is considered invalid and an interrupt of the monitor subsystem is executed with indication of the interrupt flag in the interrupt word register.

Twelve percent of all the equipment in the multiprocessor with maximal configuration is covered by on-line checking hardware.

On-line checking of the rest of the equipment is performed by a special task started upon request by the monitor subsystem. Execution time for this task does not exceed 3 ms; interval before start of task is set by the programmer. Task operation generates a monitor subsystem message on whether or not the multiprocessor is in working order.

Functional testing software is oriented to checking architectural capabilities of the multiprocessor and supports detecting and localizing malfunctions down to a functional assembly with no direct tie-in to the equipment being checked.

Functional tests are divided into tests executed in the monitor subsystem and those executed in the multiprocessor. The first group of tests checks proper functioning of the multiprocessor control unit nucleus under conditions of a microprogram halt. Execution time for this group of tests is determined by monitor subsystem speed. The second group of tests is executed directly in the multiprocessor.

A malfunction detected by functional test software is localized by hardware-software facilities. The search method is oriented to finding the faulty equipment which caused a particular functional error and outputting messages on faulty circuit name and coordinates to service personnel. Fault localization precision allows repair by personnel with average skills.

The PS-2000 multiprocessor can be operated only with the monitor subsystem (SM-2 minicomputer). The monitor subsystem should be tasked with the functions of searching for faults in the multiprocessor; the multiprocessor should be considered an object of checking with respect to the control computer.

To enable communication between the multiprocessor equipment being checked and the monitor subsystem, a network for polling multiprocessor equipment checkpoints within the multiprocessor has been incorporated. Used as checkpoints are outputs from the majority of registers, counters and flipflops, and a number of outputs from combinatorial circuits. The network for polling checkpoints in processing devices is made with elements with storage and organized as a bus. The network for polling checkpoints in control devices is made as a tree of switches with output to the diagnostic bus, to which the lines for the processing devices and checkpoints of a number of control device registers also output. There are 7,500 checkpoints in a fully configured multiprocessor.

The checkpoint polling network is controlled by using special control interface commands. Operations for changing the status of diagnostic flipflops are also executed through the control interface.

Diagnostic flipflops in the fault search mode disrupt feedback (cross communication) of automata and issue additional control actions to automata and combinatorial circuits; this facilitates checking their working order.

Asynchronous circuit (micro instruction decoder) output states are written to registers in the processing device checkpoint polling network; this allows checking asynchronous circuits in the dynamic mode. Synchronous automata circuits or combinatorial circuits for generating strobes with short length are checked in the static mode. To achieve the static mode of operation, the multiprocessor clock generatopr is controlled by special interface commands sent from the monitor subsystem. The static mode of operation allows finding persistent and constant "breaks" or "shorts".

A check is made in the dynamic mode to see that the total value of time delays in the microcircuits does not exceed the value sufficient to ensure serviceability of a given circuit assembly. Further search of a microcircuit with deteriorated dynamic parameters must be made by using an oscillograph and special debugging microprograms.

The multiprocessor checkpoint polling network allows localizing a fault down to a circuit assembly located between a point, the state of which does not match the reference value, and points, the states of which have been correctly generated in this check. The size of a faulty circuit assembly localized by the fault search on average does not exceed 20 integrated circuits. A faulty assembly is identified by board name and name of signal generated incorrectly.

Diagnostic equipment built into the multiprocessor is rather simple; this affords high reliability of the "diagnostic nucleus," the working order of which is a necessary condition for executing any check mode.

Fault localization software controls the multiprocessor checkpoint polling network and diagnostic flipflops.

PS-2000 diagnostic software structure is shown in the drawing. Test nomenclature includes software for checking the working order of the control unit, processing unit and I/O channel.

The PS-2000 diagnostic and test system is stored on external storage in the monitor subsystem. For execution, tests are loaded into the SM-2 main memory. Checks made at program level predominate in the system. Microprograms for checking multiprocessor working order are included in the diagnostic tests and are loaded into microinstruction memory during test execution. Tests are performed under control of the organizing program-dispatcher.

Two standard modes for fault localization, interactive or automatic, have been implemented in the system.

In the interactive mode, the diagnostic test system is controlled by the operator. A specific diagnostic test or group of tests combined by one character file specification may be started for execution. The dispatcher instruction

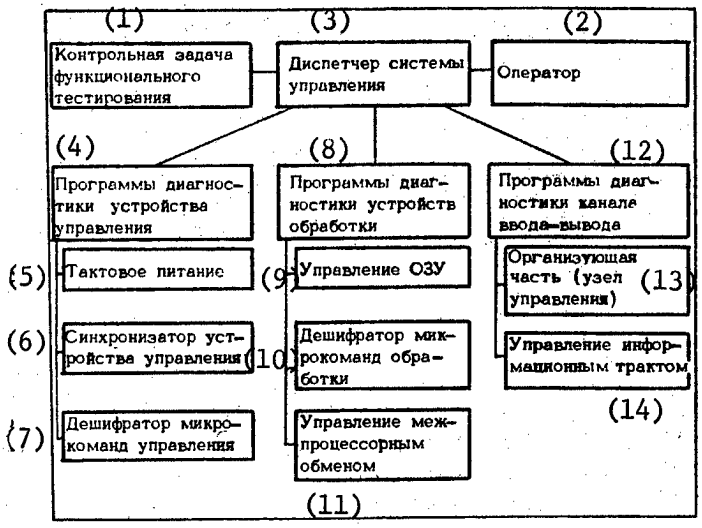


Fig. Structure of PS-2000 diagnostic software.

Key:

1. functional testing check task
2. operator
3. control system dispatcher
4. control unit diagnostic programs
5. clock supply
6. control unit synchronizer
7. control micro-instruction decoder
8. processing unit diagnostic programs
9. main memory control
10. processing micro-instruction decoder
11. interprocessor exchange control
12. I/O channel diagnostic programs
13. organizing part (control assembly)
14. information path control

set allows forming the required sequence of execution of diagnostic tests and formatting them as jobs on external storage in the monitor subsystem. In the interactive mode, the dispatcher enables displaying the existing composition of the diagnostic test system (catalog of tests and catalog of jobs are printed out). Faults are localized in the interactive mode by sequential calling of diagnostic tests for execution.

In the automatic mode of the diagnostic test system operation, faults detected by functional tests are looked for without operator intervention. If a functional test detects an error, a message is generated on the operator console and control is passed to the diagnostic test system dispatcher. After analyzing the number of the error detected by the functional test, the dispatcher calls for execution of the appropriate diagnostic test or group of tests.

The structure of each diagnostic test is a model of the equipment to be checked, specified as logical expressions or in table form. Thus, for example, the operating assemblies of the PS-2000 control and processing units, pertinent to the complex combinatorial circuits, are described by Boolean functions. For automata circuits, tabular description of the relation between output and input signals is used. Diagnostic tests generate sets of control actions in accordance with the algorithm for operation of the equipment to be checked. Each test is oriented to checking execution of a subset of hardware functions in the PS-2000 multiprocessor. Order of execution of checks in a test allows covering the equipment being checked by the method of "expanding areas." Composition of a group test is selected according to the same principle so that a particular functional assembly is checked by sequential execution of the required set of diagnostic tests.

The dispatcher in the PS-2000 diagnostic test system supports displaying the algorithm for a specific check, which detected the error, on the operator console of and allows executing one-time repetition or recycling of a given check. The PS-2000 disk diagnostic test system operates under control of the ASPO SM EVM operating systems.

This PS-2000 diagnostic test system operates best in diagnosing persistent failures. Using the hardware-software diagnostic facilities in the fault search mode yields a mean time to recovery not exceeding two hours for the PS-2000 multiprocessor.

8545
CSO: 1863/80

DISPLAY PANELS FROM ARMENIA

Moscow EKONOMICHESKAYA GAZETA in Russian No 4, Jan 84 p 24

[Text] ArSSR (TASS)--A branch of the factory "Elektron" in Kafan, this enterprise produces assemblies, parts for computers, display panels and other electronic goods. It is performing shock work in 1984. A new technique is being successfully adopted here. Brigades working on a single job authorization are being set up. Labor and production discipline is being reenforced. The photo shows one of the leading adjustors of the factory "Elektron", I. Vardanyan, debugging a display panel [photo not reproduced].

9992
CSO: 1863/121

UDC 681.3.?

ELEKTRONIKA-60 - DIGITAL INTEGRATOR COMPUTER COMPLEX

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
(manuscript received 6 Jan 83, after revision 5 May 83) pp 44-49

[Article by Anatoliy Yakovlevich Drovyannikov, engineer, Main Information and Computing Center, KaSSR Ministry of Consumer Services (Alma-Ata); Anatoliy Stepanovich Kutovoy, engineer, TRTI [Taganrog Radio Engineering Institute] (Taganrog); Oleg Borisovich Makarevich, doctor of engineering sciences, TRTI (Taganrog); and Vladimir Vasil'yevich Plotnikov, candidate of physico-mathematical sciences, Dnepropetrovsk Metallurgical Institute (Dnepropetrovsk)]

[Excerpts] As is known [1], throughput in a general-purpose computer can be increased by connecting a problem-oriented processor to it. Such a computer complex has rather extensive capabilities with a considerable increase in throughput compared to the initial computer. The main amount of computations in it is effected in the parallel processor, while the computer effects data preparation and processing, analysis and qualitative assessment of intermediate results, and solving the problems of managing data exchange and dispatching. In the process, the output information from the parallel processor can be fed directly to controlled objects, bypassing the computer. This is how the Elektronika-60 microcomputer - digital integrator (UTsI) computer complex is organized.

This integrator is designed to expand the functional capabilities of the microcomputer, which also fully determined its structural design (fig. 1) [not reproduced] and functioning [2].

This integrator has 16 digital integrators and a fully accessible switch which allows effecting transmission of information between the inputs and outputs of all the integrators without restrictions; it also has devices allowing information exchange with the computer and synchronization of the operation of the entire unit.

Digital Integrator Main Specifications

Number of digital integrators:	16
Size of registers for integrands and remainders:	16 bits
Switching of digital integrators:	electronic fully accessible time
Integration step duration:	128 microseconds

Equivalent speed computed for add operations for one digital integrator:	about 500,000 ops/sec.
Coding of information:	binary
Number of outputs for DAC:	2
Number of outputs for step actuating mechanisms:	3
Size of input information:	70 bytes
Time for recording input information:	not over 4.5 ms
Element base:	K502, K164 integrated microcircuits
Dimensions:	280 x 240 x 10
Power consumption:	not over 4 W

This complex of the microcomputer and digital integrator can meet various requirements for a broad range of users because of its rather high hardware flexibility and advanced software.

BIBLIOGRAPHY

1. Samoylov, V. D.; Aristov, V. V.; and Tarasenko-Zelenaya, L. I., "Non-homogeneous Computer-Measuring Systems with a Central Interpreting Small Digital Computer," in "Neodnorodnyye vychislitel'nyye sistemy" [Non-homogeneous Computer Systems], Kiev, Naulova dumka, 1975, pp 77-97.

2. Kalyayev, A. V., "Teoriya tsifrovykh integriruyushchikh mashin i struktur" [Theory of Digital Integrating Machines and Structures], Moscow, Sovetskoye radio, 1970, 472 pages.

8545
CSO: 1863/80

UDC 681.12

DEVICE FOR MATCHING ELEKTRONIKA-60 MICROCOMPUTER TO A318-6 PERIPHERAL MEMORY CASSETTE MODULE OF SM COMPUTERS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 (manuscript received 11 May 83) pp 21-23

[Article by L. I. Skhodtseva and I. V. Chirkin]

[Excerpts] Cassette tape units have become widespread during the past few years as the peripheral memory of mini- and microcomputers. The cassette tape used in these units is convenient to handle, has high capacity and is characterized by low storage cost per data bit. One of the main advantages of the cassette tape unit is the capability of replacing paper tape input-output devices, traditional for mini- and microcomputers, by it, which permits one to realize a cassette operating system that is much faster than a paper tape system on these computers due to the high input-output feed and also automation of search, read and write of data files.

The nomenclature of cassette tape units produced by Soviet industry and CEMA countries unfortunately does not permit the use of any of them in the peripheral devices of the Electronika-60 microcomputer, widely distributed in the USSR. None of the produced cassette tape units is matched to the channel of this computer.

The described matching device (US) permits introduction of peripheral memory cassette modules of type A311-4 of SM computers, A318-6 of SM computers and others having an interface port of 2K, as peripheral equipment for the Elektronika-60 microcomputer.

The A311-4 and A318-6 modules for the SM computer are included in the set of the Akkord-06 data preparation device, which permits these cassette tape units to be used both independently of the computer and jointly with it. The cassette tape unit in this device is designed for preparation and correction of data arrays. The cassette tape unit, together with the unit, performs:

initial loading of the main memory of the computer;

reception of data from the processor and writing of it onto magnetic tape;

retrieval, reading and transmission of the desired data to the processor.

The system support consists of a drive that permits the cassette tape unit to be used as an operating system.

Structurally, the matching device is based on a printed-circuit card of type B in designs of UTKSVT [not further identified] using TTL [transistor-transistor logic] microcircuits of medium-scale degree of integration. The card is installed on the seventh installation point of the A318-6 module of the SM computer. The matching device is connected to the computer channel by a KPPR flat cable (up to 3 meters long) and by the standard connector of the Elektronika-60 microcomputer. The connector is installed in the free space of the computer chassis with regard to the necessary priority level of the cassette tape unit.

6521
CSO: 1863/122

UDC 681.3.06.62

COMBINING LSI DIGITAL CIRCUITS FOR FFT PROCESSORS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 20 Apr 82, after revision 22 Apr 83) pp 12-17

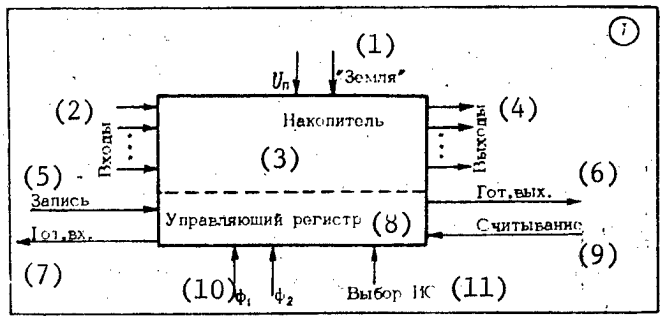
[Article by Vadim Ivanovich Veshnyakov, engineer; Mikhail Dmitriyevich Kardashchuk, candidate of engineering sciences; and Vladimir Fedorovich Koval', engineer; all from the SKB MMS IK AN USSR [Special Design Bureau for Mathematical Machines and Systems, UkSSR Academy of Sciences Institute of Cybernetics](Kiev)]

[Text] A large number of IC's have recently been developed for digital processing of signals; these may be divided into two groups. In the first group are single-chip universal signal processors which are complex subsystems, such as the AMI S2811, the Bell Labs DSP and others [1, 2]. Typical features of signal processors are the presence in them of an array multiplier; adders and time registers; program, RAM and ROM memories, and a program control unit. Signal processors can perform primary FFT processing of signals, digital filtration, and some types of secondary processing for which an additional control microprocessor is sometimes needed [2]. Since the FFT computation process in these processors is iterative, i.e. performed in several sequential stages, they cannot be used as the basis for high-speed, high-throughput devices; they are used only for processing lower frequency signals (speech signals, in particular [1-4]).

In the second group are simpler IC's: digital multipliers, microprocessor LSI arithmetic expander circuits, FIFO and register memories, ADC's, LSI correlator circuits, analog filters and others. By architectural-structural optimization, particularly by making a parallel computing structure, these circuits can be used to achieve higher speed and throughput in FFT processors.

FIFO memory is a serial static memory [5-8] with a register accumulator and a control register (fig. 1). The FIFO memory has parallel data inputs and outputs, inputs for the "Read" and "Write" control signals and outputs for "Ready for Input " and "Output Ready" which indicate, respectively, the memory is not full or there is information data in it. The memory can also have inputs for external timing and the "IC Select" input, upon the signal of which the outputs are in the active or high impedance (third) state. Only one type of LSI FIFO memory is produced in our country at present [6].

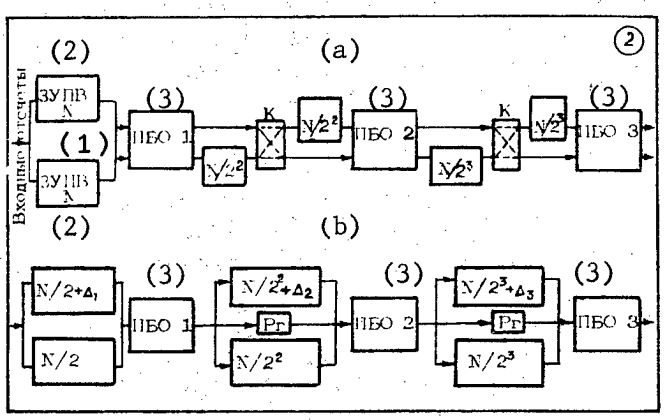
A unique property of FIFO memory is the capability of independent asynchronous access through input and output with automatic compacting of data inside; this makes it convenient for buffering input readings of a signal in the FFT processor and for rearranging data in each phase (iteration) of transformation. It is shown in [9] that FIFO memories have certain advantages compared to register memories both for single- and multi-processor pipeline computers. In the latter, FIFO memories allow loading input readings of a signal under various conditions (continuously or with spaces), continuous loading of all base operation processors (PBO), using simple addressless control which is convenient to implement by hardware, and making the entire multisection memory for a pipeline processor from just two type sizes of FIFO memory. In FFT processors to the base four or more, the presence of FIFO memories yields a gain in memory size. In FIFO memories in principle, one can build a pipeline processor with any type of algorithm (for example, with true order of data at input and output), but in doing so, FIFO memory organization is complicated; thus, the canonical graph is the most convenient.



Functional diagram of FIFO memory

Key:

1. "ground"
2. inputs
3. accumulator
4. outputs
5. write
6. output ready
7. input ready
8. control register
9. read
10. F_1, F_2
11. IC select



Organization of pipeline FFT processor
a) with register memory
b) with FIFO memory

Key:

1. input readings
2. RAM
3. base operation processor

Designing similar processor memory with RAM IC's is difficult because of the need for address control and impossibility of implementing in it simultaneous read and write. As a rule, RAM IC's are also single-charged and therefore inconvenient in designing.

Shown in fig. 2a is the organization of a pipeline processor with register memory. Included between the i-th and (i + 1)-th base operation processor are two serial memories with shift registers with a capacity of $N/2^{i+1}$ each, and between them switch (K) of two input channels per two outputs. Let us

note that for continuous operation of all base operation processors in this processor, buffering of input readings to continuously load base operation processor 1 is still needed. For this, we need another two RAM's with a capacity of N words each. This memory organization, although recognized as expedient [3, 10], imposes restrictions on the base operation processor structure: The latter must have an input and output per two parallel complex operands to receive and output them simultaneously. This structure of base operation processors is inefficient because of the large number of links and complexity. Processors with FIFO memory do not have this shortcoming since there is a base operation processor input and output for one complex number.

Organization of a pipeline FFT processor made of these base operation processors and FIFO memories is shown in fig. 2b. Each memory section has upper and lower FIFO memories and a buffer register (Pr) connected linked through inputs and outputs by bus lines linking them to the base operation processors. Each upper FIFO memory has additional memory Δ_i (where i is the memory section number). The necessity of it stems from situations occurring when the upper FIFO memory filled with $N/2^i$ numbers must be read and filled at the same time [7]. The processor performs an N-point FFT to the base two with time slicing and binary-inverse order of data at output (canonical graph). For other algorithms, more complex organization of FIFO memory with increased capacity is required.

In each processor section, a pair of input operands is selected from two FIFO memories (or one FIFO memory and one buffer register), and similarly, a pair of output operands is written to both FIFO memories (or to one FIFO memory and one buffer register). Distribution of input and output operands in some FFT algorithms in a similar pipeline processor is discussed in [9]. FIFO memory asynchronism allows effecting the modes of operation needed for FFT implementation: Operands are either only written, or only read, or are simultaneously written and read to/from FIFO memory.

Modern microprocessor hardware implementing the FFT base operation are the LSI arithmetic expander circuits K587 IK3, U832-K1883 VR2 and K588 VR1 [11] oriented to digital processing of signals. These LSI circuits have an array multiplier, adder, program control unit, general-purpose registers, two-way data buffers, search and normalization circuits, an autonomous synchronization circuit and some other special assemblies. Universality inherent to microprocessors has been achieved in these circuits, but their complexity predetermined applying the technology of CMOS or n-channel MOS structures; thus, these circuits have relatively low speed.

Faster base operation processors can be made from less complex, but faster bipolar IC's in series KR1802, KR1804, K589 [12], K1813 and K531, multipliers, multipliers with an adder-accumulator, ALU sections, registers, bus shapers and multiplexers. The structure of these base operation processors is also convenient for implemention as an LSI single-chip circuit. With that, besides a considerable reduction in number of IC packages needed for a multiprocessor FFT, the speed of the base operation processors is enhanced. The latter is due to the basic delay of the signal in the LSI circuit being accounted for by

the input/output buffers; therefore, implementing a pipeline computer structure on an LSI single-chip circuit yields advantages over a similar structure made of several IC's.

The base operation processor structure or its component, the processor element (PE), which implements the function

$$Y_{1,2} = X_1 \pm WX_2 \qquad (1)$$

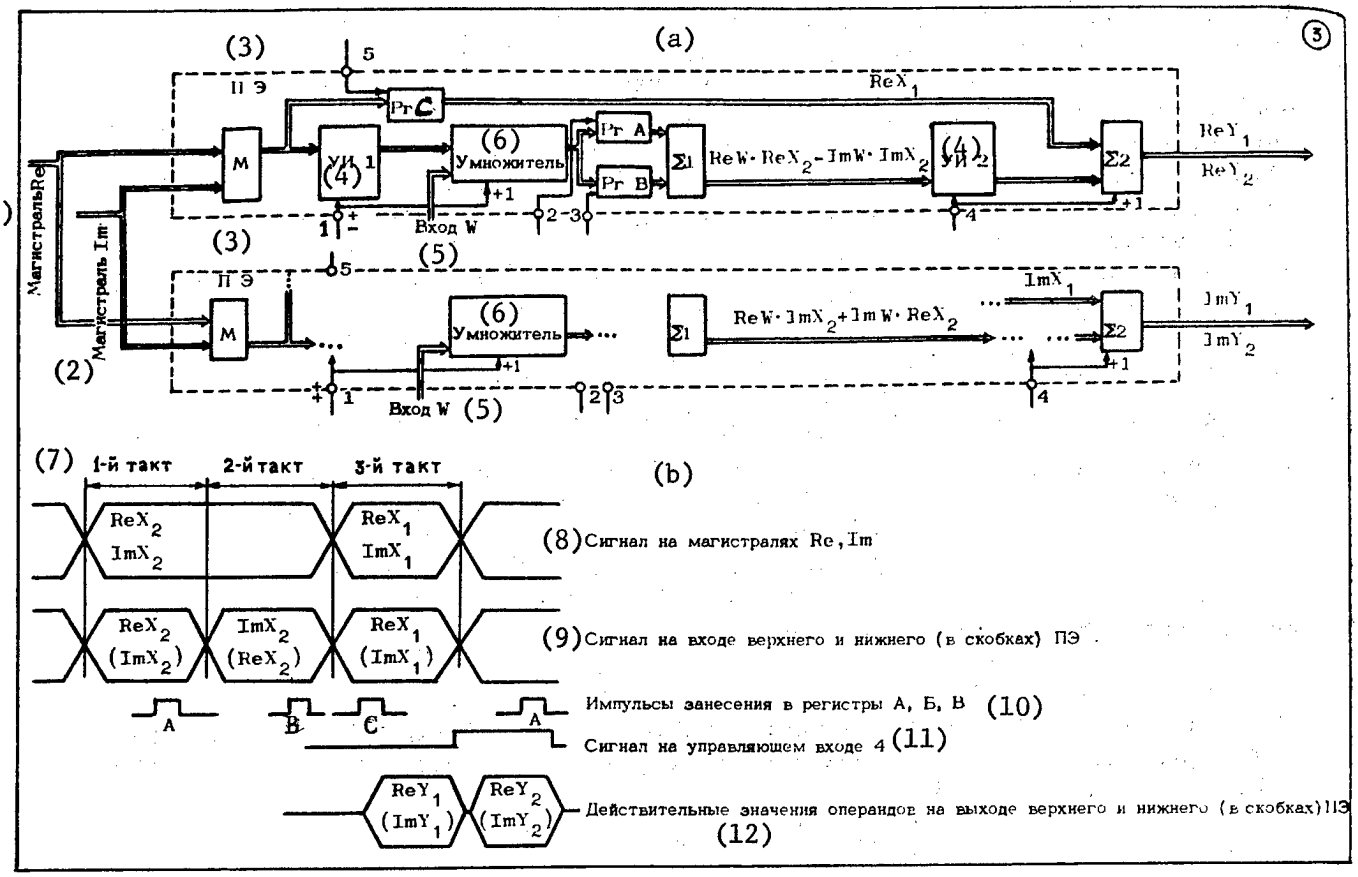
where $X_{1,2}$ and $Y_{1,2}$ are the input and output complex operands, respectively, and W is the complex coefficient, must meet the following requirements:
--The processor element structure must be optimized by number of external leads;
--The processor element with small word size must afford acceptable FFT precision through the capability of computations with both fixed and floating point;
--To attain high speed, the computer structure must be the pipeline type, and all arithmetic and functional units must be the parallel type;
--The processor element may contain only one multiplier since the latter is the bulkiest structural assembly, but with that, there must be the capability of parallel processing of the base operation with two or more processor elements.

To multiply complex numbers, one can use the functional scheme [13] which computes the product by two multiplications of real numbers. But a complex multiplier in this case is inconvenient, first, because of the necessity for simultaneous loading of two complex operands (and accordingly, a large number of leads), and second, because of the fact that in array form, it contains a full array of adders while in the multiplier there is the capability of optimizing the array structure, for example, by the (Booth) algorithm or by the algorithm with analysis of two bits [15], where the adder level quantity is halved (partial products).

Shown in fig. 3a is a base operation processor structure implementing function (1) and consisting of two processor elements and two input multiplexers (M). Each processor element has an input for an operand, a real or imaginary part of the mantissa of the complex number X_1 or X_2, and one for the W factor.

Input operands are given in two's complement code with the point fixed after the sign bit, and the factor, by a positive number with a fixed point (the signal at control input 1 plays the role of the sign). The operand input is connected to an array multiplier through controllable inverter UI1, through which the input operand goes to the multiplier input in true or inverse code. Multiplier output word size equals that of the input operand. At the multiplier output, there are two buffer registers Pr A and Pr B with synchro-inputs 2 and 3. Outputs from these registers are connected to the adder Σ1. Also, buffer register Pr C with synchro-input 5 is at the processor element input. Inverter UI2 controllable through input 4 is at the adder 2 input.

In the first step (fig. 3b), operand ReX_2 goes to the upper processor element input, and ImX_2 goes to the lower one. These operands



Processor element structure (a) and timing diagram of its operation (b)

Key:

1. Re bus
2. Im bus
3. processor element
4. controllable inverter 1, 2
5. W input
6. multiplier
7. step 1, 2, 3
8. signal on Re, Im buses
9. signal at input of upper and lower (in parentheses) processor elements
10. pulses entered in registers A, B, C
11. signal at control input 4
12. real values of operands at output of upper and lower (in parentheses) processor elements

M - multiplexer
Pr - register A, B, C

go without inversion to the multiplier inputs. At the same time, the multiplier ReW goes through the W input in both processor elements; at the end of the step, the respective products are written to the registers Pr A from the multiplier outputs.

In the second load step, ReX_2 goes to the lower processor element input, and ImX_2 goes to the upper one. The signal at control input 1 in the upper processor element is inverted, as a consequence of which the operand goes with inversion to the multiplier input, and the carry bit goes to the low-order bit of the multiplier through the "+1" circuit. At the same time, ImW is sent through the W input of both processor elements, and the respective products are written to the registers Pr B. After this, adder Σ 1 in the upper (lower) processor element generates the difference (sum) of the paired products (see fig. 3a), and this value goes without inversion to the adder 2 input.

In the third step, ReX_1 goes to the upper processor element input, and ImX_1 goes to the lower one; at the start of the step, these operands are written to the registers Pr B. Generated within the summing delay time at the output of the upper processor element is the real value of the output operand

$$ReY_1 = ReX_1 + ReW \times ReX_2 - ImW \times ImX_2,$$

and at the output of the lower one

$$ImY_1 = ImX_1 + ReW \times ImX_2 + ImW \times ReX_2.$$

Then in the first step of the next cycle, the control signal at input 4 is inverted, and generated at the output of the upper processor element is the value

$$ReY_2 = ReX_1 - ReW \times ReX_2 + ImW \times ImX_2,$$

and at the output of the lower one

$$ImY_2 = ImX_1 - ReW \times ImX_2 - ImW \times ReX_2.$$

In the processor element, there is no check for overflow of the bit network, and so scaling is required. For this, the digital values from Pr A and Pr B go to the Σ 1 inputs with shifting by one bit to the right, and overflow does not occur, but the numerical result is halved. Similarly, $ReX_1 (ImX_1)$ goes to the Σ 2 input also with shifting by one bit to the right; therefore, the numerical values of the output operands are also halved, while within $\log_2 N$ iterations they will be reduced N-fold, which is also required for computing spectral components by time signal values.

Shown in fig. 4a is a possible organization of a pipeline FFT processor which performs computations with floating point with grouped order. Normalization is performed throughout the enter N-number array. When at least one positive (negative) operand contains a 1(0) in the high-order bit of the mantissa, the array is normalized. There is a high-order bit analyzer at the output of the base operation processor of each section; when all positive operands within the bounds of the N-number array contain a 0 in the high-order bit, while the negatives contain a 1, the operand mantissas, shifted one bit left, go to the next base operation processor, the sign is kept, and the grouped order is changed by 1. Consideration of the order is performed by external circuits; a shift can be implemented by either four-input (instead of two-input) multiplexers or within the processor element; the latter alternative is preferred in implementing processor elements as LSI single-chip circuits.

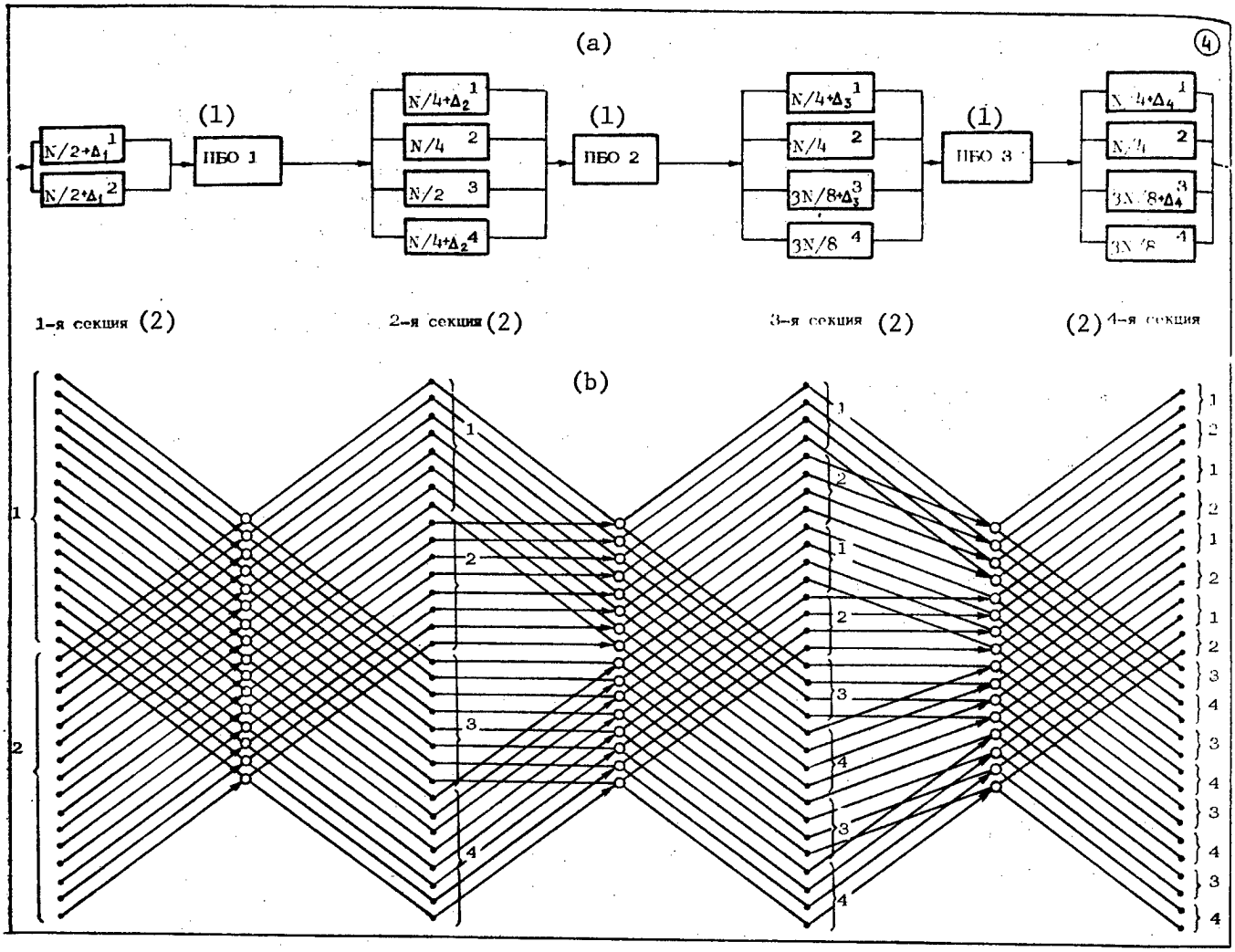


Fig. 4. Organization of pipeline FFT processor with full buffering of N-number array in each section; (a) processor structure with FIFO memory, (b) algorithm graph

Key:
1. base operation processor 1, 2, 3
2. section 1, 2, 3, 4

Parallel FIFO memory with a size of more than N complex words in each section is needed in that case for full buffering and analysis of an N-number array. In such an FFT processor, many types of algorithms [3, 10] with time slicing can be implemented. Shown in fig. 4b for example is part of an FFT algorithm graph when N = 32 with true order of data at input and output. It can be seen from the algorithm how operands are grouped in reading from FIFO memories of each section and to which FIFO memories of the next section the new operands

are written. Nominal size (none smaller is possible) of all FIFO memories is shown in fig. 4a. Let us note that the canonical graph can be implemented with dual paralleling of FIFO memories in each section [9] with a size somewhat less than their total.

The size of the total memory in the pipeline FFT processor in question is large and redundant. Designing similar memory with RAM would be difficult because of the need for address control and large quantity of IC packages. Making it with "shift-register" type storage also entails many system and design limitations [9]. Practical implementation is possible only with economic FIFO memory IC's. The large size can be reduced somewhat if normalization is performed for example through one and not in each section.

In the FFT processor, delay in deriving the final result is also increased because of full buffering, but the main parameter of throughput of the pipeline processor and thus the band of the signal to be processed is not reduced.

Processor element word size is governed by that of the input readings and size of N transformation. Let the input samples be given with a precision of r bits, and the processor element be m-bit. A one in the low-order bit with weight 2^{-m} is lost with each iteration, while within $n = \log_2 N$ iterations, the error introduced will be $n \times 2^{-m}$. So that the introduced error does not exceed the weight of the low-order bit, it is necessary that $2^{-r} > \text{or} = n \times 2^{-m}$, or $m - r > \text{or} = \log_2 n$, i.e. the processor element must have in addition m - r bits, which should be rounded at the end of the transformation. To specify factors ReW and ImW, r bits are sufficient. For example, an 8-bit sample of the input signal can be processed by a 12-bit processor when N is changed to 2^{12}.

For continuous loading of all base operation processors in the pipeline FFT processor, operands have to be sent to their inputs at the same rate, and output signals read at a different one (see fig. 3b). This would be very difficult to implement using traditional types of storage, including RAM, since two RAM read cycles and two write cycles would have to be inserted in the base operation processor cycle, and in addition, buffer registers would be needed for synchronization. Optimal loading of all base operation processors in the pipeline FFT processor is simply implemented with FIFO memories; the latter receive output operands at the rate of their arrival and output them at the rate required for the base operation processors in the next section. Fully asynchronous non-timed FIFO memories are ideal in this respect [5, 6].

If the speed of the non-timed FIFO memories [7, 8] is sufficiently high, and the phase pulse (by each of which read and write can be synchronized) rate is four or more times greater that that of the base operations, then these FIFO memories essentially will afford continuous loading of all base operation processors. In other words, FIFO memories and the processor elements are conveniently matched at the machine cycle level.

This processor element can also execute some other arithmetic operations with complex operands in two's complement code. Shown in fig. 5a is a diagram of

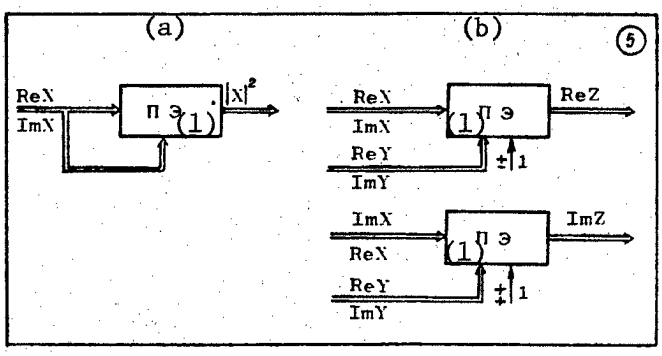


Fig. 5. Diagram of processor element in computing (a) square of modulus and (b) product of complex numbers

Key:
1. processor element

a processor element for computing the square of the modulus $|X|^2$ from the real and imaginary component, which is used, for example, in computing an energy spectrum. Operand and W factor inputs are combined; loading is performed in two steps: first ReX, then ImX. Register C is permanently set to 0, and the value of $|X|^2 = (ReX)^2 + (ImX)^2$ is generated at output.

Multiplication of the complex number $Z = X \cdot Y$ is used, for example, in computing high-speed convolution by the FFT method [4], and is executed in a way similar to the base operation in question by two processor elements (fig. 5b). In the first load step, ReX goes to the operand input in the upper processor element, ImX to the input in the lower, and ReY goes to the W input in the upper and lower processor elements. In the upper and lower processor elements, the control signal from input 1 has a "+" value. In the second step, conversely, ImX goes to the input in the upper processor element, ReX to the input in the lower, and ImY goes to the W input; in the upper processor element, the control signal from input 1 is inverted. Register C is permanently set to 0; then the value of ReZ = ReX x ReY - ImX x ImY is generated at the output of the upper processor element, and ImZ = ImX x ReY + ReX x ImY at the output of the lower.

Since the processor element computes an $a \pm b \cdot c \pm d \cdot e$ function type, it may find other applications in digital processing of signals, for example, for accumulating the sum of paired products.

BIBLIOGRAPHY

1. (Posa, J.), "Digital Processors of Analog Signals: A New Direction in Integrated Technology," ELECTRONICS, No 4, 1980, pp 93-96 [Russian translation].

2. (Blasko, R.), "U-MOS Device Connectable to Microprocessor for Processing Signals in Real Time," ELECTRONICS, No 18, 1979, pp 38-50 [Russian translation].

3. (Rabiner, L and Gould, B.), "Theory and Application of Digital Processing of Signals," Moscow, Mir, 1978, 848 pages.

4. (Oppenheim, E.), "Application of Digital Processing of Signals," Moscow, Mir, 1980, 550 pages.

5. Veshnyakov, V. I. and Kardashchuk, M. D., "Structural-Schematic Implementation of LSI FIFO Memories," MIKROELEKTRONIKA, Vol 11, No 5, 1982, pp 430-440.

6. Kogan, A. L.; Kolosovskiy, A. V.; and Lis', V. D., "K1002 IR1 LSI Register Memory," ELEKTRONNAYA PROMYSHLENNOST', No 1, 1982, pp 14-16.

7. Veshnyakov, V. I.; Korneychuk, V. I.,; and Solomonov, V. V., "Synthesis of Serial FIFO Memory," AVTOMATIKA, No 2, 1979, pp 65-69.

8. Veshnyakov, V. I.; Kardashchuk, M. D.; and Solomonov, V. V., "USSR Patent 847372 for Shift Register," published in B. I. [BYULLETEN' IZOBRETENIYA], No 26, 1981.

9. Veshnyakov, V. I. and Kardashchuk, M. D., "Bunkernoye ZU kak problemno-oriyentirovannaya BIS dlya vychisliteley BPF" [FIFO Memory as a Problem-Oriented LSI Circuit for FFT Computer], Kiev, UkSSR Academy of Sciences Institute of Cybernetics, 1982, 32 pages.

10. Bakhtiarov, G. D. and Tishchenko, A. Yu., "Implementing Devices for Digital Processing of Signals Based on FFT Algorithms," ZARUBEZHNAYA RADIOELEKTRONIKA, No 9, 1975, pp 71-97.

11. Borisov, V. S., "Mikroprotsessornyye komplekty integral'nykh skhem" [Microprocessor Families of Integrated Circuits], Moscow, Radio i svyaz', 1982, 190 pages.

12. Berezenko, A. I.; Koryagin, A. N.; and Nazar'yan, "Mikroprotsessornyye komplekty povyshennogo bystrodeystviya" [Microprocessor Families with Enhanced Speed], Moscow, Radio i svyaz', 1982, 170 pages.

13. Semotyuk, M. V. and Boyun, V. P., "Operating Devices for Summing ?Paired Products and ?Multiplication of Complex Numbers," USiM [UPRAVLYAYUSHCHIYE SISTEMY I MASHINY], No 3, 1978, pp 34-38.

14. (Wizer, C.), "High-Speed Digital Multiplier for Real-Time Signal Processing," ELECTRONICS, No 20, 1977, pp 40-49 [Russian translation].

15. Kartsev, M. A., "Arifmetika tsifrovykh mashin" [Digital Computer Arithmetic], Moscow, Nauka, 1969, 576 pages.

8545
CSO: 1863/80

UDC 621.3.049.77.019.3

COMPUTER-AIDED ANALYSIS OF LSI CIRCUIT PARAMETRIC RELIABILITY

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 15 Feb 82) pp 18-20

[Article by Vladimir Vladimirovich Gorovoy, engineer (Minsk); Aleksandr Grigor'yevich Sakhashchik, engineer (Minsk); and Sergey Timofeyevich Khvoshch, candidate of engineering sciences, LETI [Leningrad Electrical Engineering Institute] (Leningrad)]

[Text] Large-scale integrated (LSI) circuits are now widely used in various types of radioelectronic equipment; this imposes more and more stringent requirements on LSI cicuit reliability. Continuing efforts to enhance reliability have allowed establishing the following [1, 2]:
--The current standard in LSI circuit reliability is a failure rate of no more than 10^{-6} - 10^{-8} h^{-1};
--LSI circuit reliability is largely determined by parametric failures caused by the occurrence of various physicochemical degradation processes in the operating circuit.

In such a situation, developing accelerated methods of assessing LSI circuit parametric reliability is of significant value to sound application of LSI circuits and purposeful efforts to further raise the reliability standard. Analysis of theoretical and experimental efforts underway allows identifying two main directions in developments:
--application of special test modes allowing substantial acceleration in degradation processes;
--use of methods for predicting test results on computers.

The first direction requires, as a rule, developing special test equipment for high temperature tests.

The second direction is being evolved more successfully now since it allows managing with series equipment now in production to perform the reliability tests needed to obtain source data for subsequent prediction with computers.

When prediction methods are used, the following are resolved sequentially:
--synthesis of a model of degradation of distributions of fitness parameters and criteria (PKG) for LSI circuits (theoretical concepts of fitness parameters and criteria drift or experimental results are used);

--computer simulation of the process of reliability tests of a large number of LSI circuits based on the synthesized model; assessment and analysis of reliability characteristics.

In this work, the prediction method is used to assess the parametric reliability of the K583KhL1 LSI circuit. This microcircuit is a universal switching element that is part of the standard microprocessor family with low power based on injection logic, which is intended for developing computers and automated devices for broad applications.

To obtain information on the nature and rate of drift of fitness parameters and criteria of the LSI circuit, reliability tests were performed on 40 circuit samples at rated electrical load and temperature of T = 398°K. Duration of the tests was 5,000 hours. All controllable electrophysical parameters for the LSI circuit were taken as fitness parameters and criteria. The fitness parameters and criteria were checked at the start of the tests and every 100 hours. A list of all fitness parameters and criteria, their maximum tolerable values, ratings of mean values X_m and standard deviations σ at the start of the tests is shown in table 1.

Table 1. Fitness parameters and criteria for LSI circuit K583KhL1

Fitness parameters and criteria	Max. value allowed	X_m	σ
Antiring diode blocking voltage U_g, V	< or = 1.5	0.900	0.0267
Logical one output voltage U^1_{out}, V	< or = 0.4	0.1297	0.0109
Injection voltage U_{inj}, V	> or = 1	1.53	0.071
Logical zero input current I^0_{in}, micro A	< or = 200	102.1	22.4
Logical zero output current I^0_{out}, micro A	< or = 450	294.7	53.4
Input breakdown current $I^0_{in\ b}$, micro A	< or = 400	294.6	53.5

Test results were processed as in [3]. In the first phase, the relation between time of tests t and fitness parameters and criteria values was found. Linear correlation analysis was used for this [4]. Results of computation of correlation and determination factors confirm the presence of the time relation, significant and rather close to the functional, for all fitness parameter and criteria values. Then by using one-factor analysis of variance [4], the presence of a linear functional relation for all fitness parameters and criteria was verified and established. A check was made for significance level of $\alpha = 0.05$. Thus, it was shown that the change in mean values of all fitness parameters and criteria in time can be approximated by the equation

$$x = a + b \cdot t. \qquad (1)$$

The least squares method [5] was used to assess factors a and b in equation (1) and their standard deviations σ_a and σ_b (table 2).

Table 2. Assessment of values of a, b, σ_a and σ_b

Fitness parameters and criteria	a	b	σ_a	σ_b
U_q	0.8983	4.781×10^{-3}	4.328×10^{-3}	3.516×10^{-4}
U_{out}	1.1307	3.217×10^{-3}	5.438×10^{-3}	2.713×10^{-4}
U_{inj}	1.522	-6.271×10^{-3}	1.637×10^{-3}	1.713×10^{-3}
I^0_{in},	100.1	2.117	0.1213	0.1937
I^0_{out},	297.4	2.214	0.3764	0.2135
$I^0_{in\ b}$,	296.2	2.383	0.3681	0.2238

Thus, statistical analysis of the experimental results of the reliability tests allowed finding the presence of the time drift of the fitness parameters and criteria and determining the drift nature and rate.

Results from the experiments were used as source data for statistical modeling of reliability tests. Modeling was performed on the YeS1022 computer as follows. By using a random number generator and in accordance with the specified values of b and σ_b, we generated values for the rate of drift of the fitness parameters and criteria V_g for the LSI circuit. Then based on the known initial distribution of fitness parameters and criteria, we generated a sample with size n of values for the fitness parameters and criteria and computed the distribution of time to onset of failure for the fitness parameters and criteria being analyzed. This procedure was repeated for all the fitness parameters and criteria for the LSI circuit under study. From the set of values of times of failures of all fitness parameters and criteria, we chose the smallest which was also taken as the time to device failure as a whole. Then this computation was repeated for the specified number m of devices being tested. At the end, based on the simulation results, we computed and printed out the time relations for the reliability characteristics of the LSI circuit.

The computation for m = 100 took 40 minutes of machine time. Preliminary test computations showed sufficient convergence of the assessments of reliability characteristics using this sample. The time relations of probability of no-failure operation P(t) and failure rate λ (t) were computed for each fitness parameter-criteria and the entire LSI circuit as a whole. Results for the fitness parameters and criteria I^0_{in} and $I^0_{in\ b}$ and for the LSI circuit as a whole are shown in figs. 1-3, where the solid lines denote point estimates of the reliability characteristics and the broken lines denote the confidence interval of prediction, corresponding to 95 percent probability.

The results obtained show there is considerable difference in the parametric reliability for the different fitness parameters and criteria. For clarity and convenience of comparison, shown in table 3 are the values of 95 percent life (an indicator describing the maximum serviceability of the LSI circuit

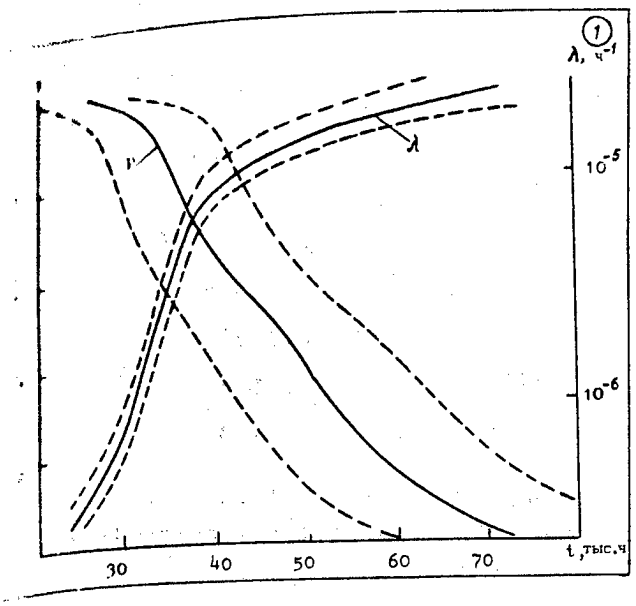


Fig. 1. Relation between P(t) and $\lambda(t)$ for I^0_{in}

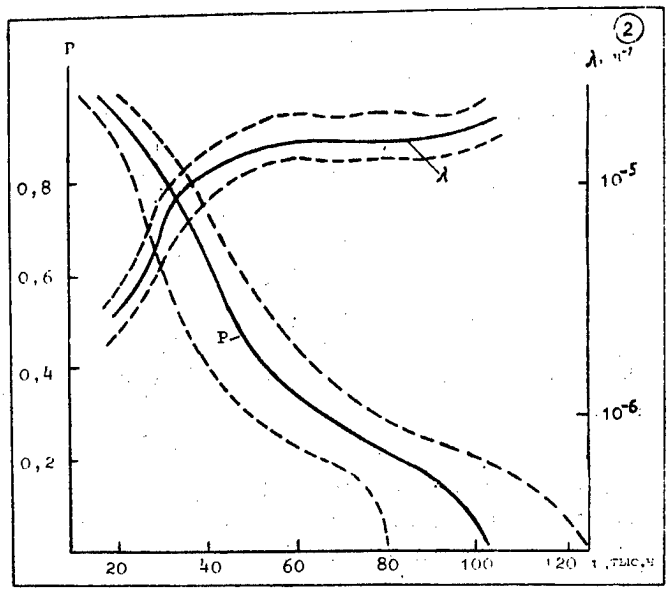


Fig. 2. Relation between P(t) and $\lambda(t)$ for I^0_{in} b

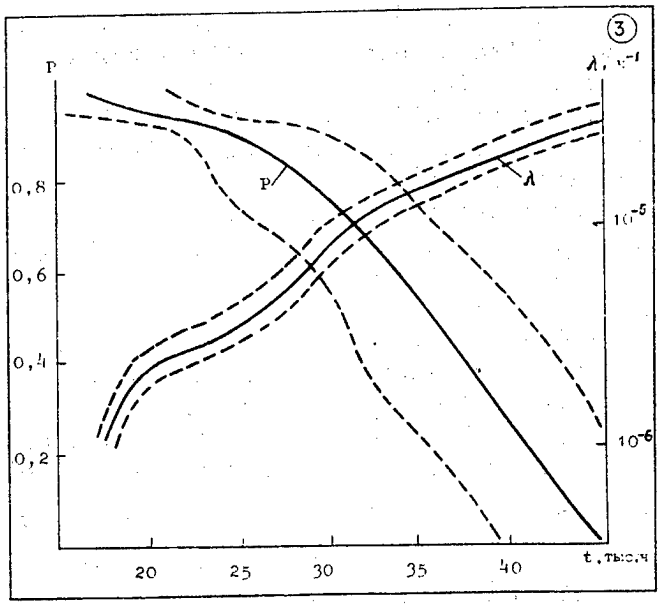


Fig. 3. Relation between P(t) and $\lambda(t)$ for the LSI K583KhL1 circuit

Key [for all 3 figs.]: t, in thousands of hours; λ, in h^{-1}

Table 3. Values of reliability characteristics by fitness parameters and criteria and the LSI circuit

Fitness parameters and criteria	95-% life, thousands of hours			Failure rate, h^{-1}		
	t_L	t_0	t_U	λ_L	λ_0	λ_U
U_d	72.3	80.7	89.0	6.9×10^{-7}	6.2×10^{-7}	5.6×10^{-7}
U^I_{out}	63.2	77.3	91.4	8.0×10^{-7}	6.5×10^{-7}	5.5×10^{-7}
U_{inj}	64.0	67.9	71.8	7.8×10^{-7}	7.4×10^{-7}	6.0×10^{-7}
I^0_{in},	26.0	32.4	38.8	1.9×10^{-6}	1.5×10^{-6}	1.3×10^{-6}
I^0_{out},	36.7	46.7	56.7	1.4×10^{-6}	1.1×10^{-6}	8.9×10^{-7}
$I^0_{in\ b}$,	14.8	18.8	22.8	3.4×10^{-6}	2.7×10^{-6}	2.2×10^{-6}
LSI circuit	14.8	18.8	22.8	3.4×10^{-6}	2.7×10^{-6}	2.2×10^{-6}

and equal to the time in which 95 percent of the circuits tested still remain serviceable) and corresponding to these times, the failure rates for each fitness parameter and criterion and the entire LSI circuit as a whole. Shown in the table are the point (t_0, λ_0), lower (t_L, λ_L), and upper (t_U, λ_U) confidence bounds.

These results show the parametric reliability level of the majority of fitness parameters and criteria of the LSI circuit studied is rather high. We now have the capability to soundly and purposefully approach developing measures to further enhance this LSI circuit's parametric reliability. Using the program developed to simulate the various possible ways of raising LSI circuit parametric reliability, it was found that the most acceptable way is to raise the technological margins of critical fitness parameters and criteria. By technological margin, we mean the difference between the tolerance and maximum value of the fitness parameter-criterion. In particular, we found that to raise the 95 percent life of the LSI K583KhL1 circuit according to parametric failures by 10,000 hours, the technological margin of the I^0_{in} parameter has to be increased by 25 microamps.

BIBLIOGRAPHY

1. Aleksanyan, I. T., "Features of Problems of Reliability of Microelectronic Products," ELEKTRONNAYA TEKHNIKA, No 6(84), 1980, pp 3-8.

2. Aleksanyan, I. T., "Methodological Principles of Simulation in the Theory of Reliability of Highly Reliable Products," ELEKTRONNAYA TEKHNIKA, No 4(90), 1981, pp 7-10.

3. Gorovoy, V. V.; Savotin, Yu. I.; and Sakhashchik, A. G., "Statistical Analysis of Parametric Reliability of LSI 256-Bit Memory Circuits," ELEKTRONNAYA PROMYSHLENNOST', No 2(98), 1981, pp 43-45.

4. Gerasimovich, A. I. and Matveyeva, Ya. I., "Matematicheskaya statistika" [Mathematical Statistics], Minsk, Vysheyshaya shkola, 1978, 200 pages.

5. Dlina, A. M., ed., "Matematicheskaya statistika" [Mathematical Statistics], Minsk, Vysheyshaya shkola, 1975, 398 pages.

8545
CSO: 1863/80

UDC 681.323

AN APPROACH TO STRUCTURAL SYNTHESIS OF MICROINSTRUCTION MEMORY BASED ON PROGRAMMABLE LOGIC ARRAYS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 18 Jan 82, after revision 15 Jan 83) pp 20-24

[Article by Vadim Abramovich Raykhlin, candidate of engineering sciences, and Moisey Isaakovich Shvartsman, engineer; both from the Kazan Aviation Institute (Kazan)]

[Excerpts] Introduction. A considerable share of the research on circuits implemented by programmable logic arrays [PLA's] has been devoted to studying the various applications of the theory of automata (conversion of tables, minimization, coding, etc.) related to mapping the circuit to a "dimensionless" matrix [1-5]. With that, minimizing chip area has been the main criterion. Less attention has been paid to the major practical problem, dealt with in this article, of designing complex digital devices based on a system of series PLA's, when the problems of decomposition of structure are foremost [6, 7]. The algorithm in this article for decompositions through outputs was first formulated by the authors in [8] and analyzed in [9]. In known systems for computer-aided design [CAD] (see [10], for example), decomposition of PLA's through outputs is accomplished the natural way.

Structural Synthesis Results. The system developed was used for comparative structural synthesis based on PLA's for two microinstruction memory devices:

--microinstruction memory for a specialized processor [12]; the source microprogram for it has 415 microinstructions and 92 microoperations. In implementing the microinstruction memory, we chose a PLA with m = 14, q = 96 and r = 8. The microinstruction memory table was divided into 12 subsystems by using the full algorithm. We managed to place the minimized strip on 19 PLA packages; there were 12 types. For comparison, the same microinstruction memory was implemented with 256 x 4-bit ROM (m = 8, r = 4); integration scale was about the same as that for the PLA selected. The number of packages needed increased to 37. With that, all the ROM's were programmed differently;

--microinstruction memory for the YeS2433 processor with 2048 x 128-bit memory, PLA's and ROM's with previous parameters. The memory as a whole was divided the natural way into 8 blocks of 512 x 64 bits each. Synthesis by using the system was performed serially block by block. As a result, implementation was obtained with 135 PLA packages. When ROM was used, 256 packages were needed.

Thus, there is a foundation for believing that implementing relatively large microinstruction memory based on PLA's allows halving, on the average, the number of packages needed compared to using ROM's with the same integration scale.

BIBLIOGRAPHY

1. Yakubaytis, E. A.; Bul', Ye. S.; Lange, E. E. et al., "Technique for Synthesizing Asynchronous Automata with PLA's," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, No 4, 1980, pp 23-31.
2. Achasova, S. M. and Bandman, O. L., "Matrix Method for Synthesizing Combinatorial Circuits and Logic Converters of Finite Automata," TEKHNICHESKAYA KIBERNETIKA, No 6, 1975, pp 99-106.
3. Achasova, S. M. and Bandman, O. L., "Designing Control Automata with LSI Circuits," in "Vychislitel'nyye sistemy. Avtomatizatsiya proyektirovaniya v mikroelektronike" [Computer Systems. Automating Design in Microelectronics], Novosibirsk, Siberian Branch, USSR Academy of Sciences, No 64, 1975, pp 26-52.
4. Zakrevskiy, A. D., "Logicheskiy sintez kaskadnykh skhem" [Logic Synthesis of Cascade Circuits], Moscow, Nauka, 1981, 416 pages.
5. Shvartsman, M. I., "Generalized Approach to Minimizing Boolean Functions," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, No 6, 1979, pp 11-14.
6. Baranov, S. I. and Sinev, V. N., "Synthesizing Automata with PLA's," USiM, No 2, 1979, pp 58-64.
7. Novikov, S. V., "Metod sinteza skhem na PLM" [Method of Synthesizing Circuits with PLA's], Kiev, UkSSR Academy of Sciences Institute of Cybernetics, 1980, 24 pages.
8. Raykhlin, V. A., "Asinkhronnyye tsifrovyye skhemy i modul'nyye struktury" [Asynchronous Digital Circuits and Modular Structures], Kazan', Kazan' Aviation Institute imeni A. N. Tupolev, 1980, 104 pages.
9. Shvartsman, M. I., "Decomposition through Outputs of Combinatorial PLA Structures," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, No 6, 1981, pp 12-17.
10. Baranov, S. I. and Zhuravina, L. N., "Universal Automated System for Synthesis of Microprogram Automata (UASSMA-YeS), USiM, No 4, 1980, pp 85-86.
11. Kravtsov, L. Ya. and Chernitskiy, G. I., "Proyektirovaniye mikroprogrammnykh ustroystv upravleniya" [Design of Microprogram Control Devices], Leningrad, Energiya, 1976, 152 pages.
12. Raykhlin, V. A.; Pavlovich, O. G.; and Shvartsman, M. I., "Problems in Developing Microprogram Control Devices Based on PLA's" in "Vychislitel'nyye i upravlyayushchiye sistemy letatel'nykh apparatov" [Computer and Control Systems for Flying Vehicles], Kazan', Kazan' Aviation Institute imeni A. N. Tupolev, 1980, pp 46-50.

8545
CSO: 1863/80

UDC 519.714.2

IMPLEMENTING SEQUENTIAL AUTOMATA WITH PROGRAMMABLE LOGIC ARRAYS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 13 Jul 83) pp 8-12

[Article by Aleksey Aleksandrovich Butov, candidate of engineering sciences, Minsk Radio Engineering Institute (Minsk)]

[Excerpts] In describing algorithms for operation of discrete control devices with large dimensions, the behavior of which has been defined only in a small part of Boolean space of input and internal variables, the language of sequences is often used as the source; it has these useful features [1]:
--its structure is close to that of natural language;
--complexity of description in this language is linearly related to the complexity of the impending circuit implementation;
--the language allows simple and unambiguous transition to a system of logic equations describing control device functions of branches and outputs.

The language of sequences is a convenient facility for describing algorithms of functioning of control devices in a broad class, including control devices in industrial automation systems [2]. In the process, signals from various transducers serve as input variables, while internal variables reflect the status of actuating mechanisms (relay coils, valves, switches, etc.).

Widely used now as the element base in synthesis of irregular control logic are programmable logic arrays [PLA]; they have well known advantages [3] compared to other types of logic modules and are adjustable bilevel array structures designed to implement systems of Boolean functions in disjunctive normal form (DNF). The PLA internal structure is shown in fig. 1 [not reproduced], where A_1, ..., A_p are the input buses; B_1, ..., B_r are the output buses; C_1, ..., B_q are the intermediate buses; and the small circles represent the locations of transistors in the PLA array field (black circles pertain to the AND array; white circles, to the OR array). Functions implemented with the PLA output buses are expressed by disjunctions of certain of the elementary conjunctions implemented with the intermediate buses.

Discussed in this article are methods of implementing control devices, the conditions of operations of which are specified in the language of sequences, with one PLA and RS, T and D flipflops.

In conclusion, let us note that by choosing the type of elements of memory, one can more fully make use of the PLA structural features, which is important in synthesizing circuits with standard PLA's having a fixed number of input, output and intermediate buses. The method of implementation based on D flip-flops assumes (when possible) use of a D register, thereby allowing reduction in the number of microcircuit packages.

BIBLIOGRAPHY

1. Artyukhov, V. L.; Kopeykin, G. S.; and Shalyto, A. A., "Nastraivayemyye moduli dlya upravlyayushchikh logicheskikh ustroystv" [Configurable Modules for Control Logic Devices], Leningrad, Energiya, 1981, 168 pages.

2. Zakharov, V. N., "Avtomaty s raspredelennoy pamyat'yu" [Automata with Distributed Memory], Moscow, Energiya, 1975, 136 pages.

3. Yakubaytis, E. A., "Programmable Logic Array Structural Synthesis," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, No 4, 1976, pp 1-10.

4. Zakrevskiy, A. D., "Logicheskiy sintez kaskadnykh skhem" [Logic Synthesis of Cascade Circuits], Moscow, Nauka, 1981, 414 pages.

5. Baranov, S. I. and Sinev, V. N., "Avtomaty i programmiruyemyye matritsy" [Automata and Programmable Arrays], Minsk, Vysheyshaya shkola, 1980, 136 pages.

6. Baranov, S. I. and Sinev, V. N., "Programmable Logic Arrays in Digital Systems (Review)," ZARUBEZHMAYA RADIOELEKTRONIKA, No 1, 1979, pp 65-82.

7. Kagan, B. M. and Kanevskiy, M. M., "Tsifrovyye vychislitel'nyye mashiny i sistemy" [Digital Computers and Systems], Moscow, Energiya, 1973, 680 pages.

8. Zakrevskiy, A. D., "Logicheskiye uravneniya" [Logic Equations], Minsk, Nauka i tekhnika, 1975, 96 pages.

9. Utkin, A. A., "Operations on Boolean Functions Produced by Substitution," in "Algoritmy resheniya logiko-kombinatornykh zadach" [Algorithms for Solving Combinatorial Logic Problems], Minsk, ITK AN BSSR [BSSR Academy of Sciences Institute of Engineering Cybernetics], 1975, pp 60-77.

8545
CSO: 1863/80

UDC 681.325

SM-1, SM-2 COMPUTER MODULE FOR DATA INPUT TO PROM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
(manuscript received 21 Jul 82, after revision 20 Oct 82) pp 113-115

[Article by Igor' Ivanovich Don'kin, senior scientific associate; Vladimir Ivanovich Kleymenov, engineer; and Lyudmila Fedorovna Belikova, engineer; all from the VNIIPT khimhefteapparatury [All-Union Scientific Research Institute of Technology of Chemical and Petroleum Equipment Manufacturing] (Volgograd)]

[Excerpts] Combinatorial logic circuits make up a considerable part of the hardware in computers and complex automation systems. Designing them is a complex problem, the optimal solution of which allows developing and making the most efficient and economical equipment.

The recent trend is to implement combinatorial logic on the basis of semiconductor ROM IC's. It is also advisable to use permanent storage devices for microcomputer ROM. In equipment made in single copies or small series and for laboratory experiments, electrically programmable ROM [EPROM] is widely used; the user inputs data to the EPROM by selective destruction of the homogeneity of an array containing fusible links [1].

A very widespread method for input of information to a PROM is burning links by dosed pulses of current. In the link burning process, addresses are serially fed to address inputs and burn pulses to outputs of corresponding bits according to a truth table specified by the user.

A PROM can be programmed with special equipment called PROM programmers. But domestic production of them is insufficient. Consequently, this equipment is expensive and in short supply.

As a rule, PROM programmers are controlled by mini or micro computers or have one in their composition [2]. When the user has a computer, it is advisable to load it with the maximum of functions, including preparation of arrays of information to be recorded, specification of PROM burning conditions, checking the information recorded in the PROM and others. And the more ROM programming tasks handled by the computer, the simpler the design of the programmer itself can be, or in other words, the simpler the module for data input to a PROM.

At the VNIIPTkhimnefteapparatury [All-Union Scientific Research Institute of Technology of Chemical and Petroleum Equipment Manufacturing] (Volgograd), a module for entering information into a PROM was developed and made as a separate peripheral controllable by the SM-1 computer. To interface it to a computer, a duplex register (DR A491-3M) is used. This is a standard assembly for connecting peripherals with the 2K interface, typical for the control computer systems M6000, M7000 (ASVT-M) and the SM-1 and SM-2 (SM EVM).

The module for entering information into a PROM (fig. 1 [not reproduced]) contains a receiving inverter register for the address and number of the bit to be burned (RgP), decoder of the number of the bit to be burned, block of write keys (BK1), read register (Rg Cht), unit for generating write pulses and strobes with an emergency pulse length limiter (UFZ), and replaceable PROM adapters. In particular, adapters have been provided for programming K155 RYe3, K556 RT4 and K556 RT5 PROM's for this module.

In the interactive mode, an operator uses a display console to enter the PROM microcircuit type (number of bits and addresses), which determines the source array dimension. According to the number of the PROM microcircuit to be programmed, specified by the operator from the display console, appropriate information used to generate control words for burning is fetched automatically from computer main memory

The numbers of the microcircuits being programmed are shown on the display screen and in the output document heading.

For ease in assembling devices containing several PROM microcircuits, the number of the microcircuit to be programmed should be put in a special marking pasted on the microcircuit package.

Before writing information to the current PROM address, the possibility of programming by each bit is checked, and when the initial information is not picked up by the information being entered, an error message is output to the display screen and log.

Future developments call for incorporating into the system automatic checking to see whether source information is stored in main memory by the initial information for a programmable microcircuit.

Good microcircuit contact in the adapter is achieved by adapter connector design and contact surface purity and is determined visually.

However, if contact is interrupted while information is entered, the presence of errors can be determined in check reading and comparing of information from the PROM to the source array.

Technical Characteristics of Module

PROM address range	not over 4K
PROM word size	not over 8
computer interface	A491-3M duplex register
range of software-specified lengths of write pulses when length of leading edge is not over 1 microsecond	25 microseconds -- 50 ms
time to emergency limit of write pulse length	70 ms
PROM types with replaceable adapters	K155RYe3, K556RT4, K556RT5
current consumption	
from +5V source	not over 0.25 A
from +12V source	pulse not over 0.1 A
dimensions of module for entering information into PROM	not over 70 x 150 x 200 mm
weight	not over 1.5 kg

BIBLIOGRAPHY

1. Gordonov, A. Yu., ed., "Poluprovodnikovyye zapominayushchiye ustroystva" [Semiconductor Memory Devices], Moscow, Radio i svyaz', 1981, 344 pages.

2. Gladkov, A. M.; Nagayets, N. V.; and Teslenko, V. A., "Programmer for Microprocessor Devices," PRIBORY I SISTEMY UPRAVLENIYA, No 7, 1981, p 23.

8545
CSO: 1863/80

UDC 681.326

SOME TRENDS IN DEVELOPMENT AND PRINCIPLES OF ORGANIZING DEVELOPMENT OF SOFTWARE FOR MANAGEMENT INFORMATION SYSTEMS (ASU)

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 11 Mar 83) pp 3-7

[Article by Boris Borisovich Timofeyev, Academician, UkSSR Academy of Sciences; Aleksandr Fedorovich Kulakov, doctor of engineering sciences; and Grigoriy Aleksandrovich Kozlik, candidate of engineering sciences; all from the Institute of Automation (Kiev)]

[Text] Modern management information systems [ASU] are complex man-machine systems made up of hardware, software, information, supervisors and ASU employees. ASU evolve through transferring to a computer an ever larger number of functions of analysis, planning and management for industry. In doing so, the functions of information acquisition and storage, planning and management are realized by using appropriate software (PO). Software complexity is increased when we develop modern organizational-technological systems for managing industry. These are characterized by a substantial increase in dimensions of the problems to be solved, a complex hierarchy of functions and criteria for optimization (suboptimization), and rigid restrictions on time, technological and economic parameters.

In connection with this, the role of ASU software is continually increasing and in ASU development, the emphasis has long ago been shifted from hardware to software. Labor and material costs for software development, operation and maintenance are increasing accordingly. In the majority of cases, these costs are substantially higher than those for ASU hardware.

Let us note the following trends in ASU software development:

1. ASU software is developed, as a rule, in two phases. First phase efforts are performed primarily in specialized scientific research institutes and design bureaus; second phase, by the joint efforts of scientific research institutes, design bureaus and the shops in the enterprises being automated (the customers). The transition from the first to the second phase is characterized by an 1.5 to 2-fold increase in the number of tasks to be handled (and accordingly the subsystems) because of in-depth elaboration of the subsystems previously developed, increase in scope of automation of activity in an enterprise, more profound and efficient consideration of the effect of managerial relations and because of an increase in the number of optimization tasks.

2. Efforts on ASU software modernization (maintenance) are intensively performed also after introduction of second-phase tasks. These efforts are performed by shops in the enterprises being automated with participation of developing organizations. Software modernization stems primarily from the following: bugs are discovered, new tasks are posed, objects and control systems are improved, new hardware is used, the sphere of software application is expanded, the information base is improved, and the need emerges for developing unified data bases (banks).

3. Development of enterprise ASU software is substantially influenced by the general trend in enterprise ASU development as a multilevel (trilevel, most often) system. Supported on the top level is technical and economic planning, management of production and sales, supply of materials and equipment, and inter-shop management of production; on the middle level, accounting, planning and management of production in the shops; on the lowest level, process control and management of work stations.

4. Further development in enterprise ASU is shaped by integration of management systems in use into the statewide system (OGAS). This integration should be performed by centralized acquisition and processing of information within the bounds of the unified system for territorially distributed data banks.

5. In connection with complication of software and demands for efficient operation of it in a given computer environment (hardware and systems software), the value of formal methods and automated program-algorithm facilities for design and modernization of software packages increases.

Since the functions of analysis, accounting, planning and management are directly implemented in software, it has a decisive effect on ASU efficiency.

Let us consider some ways of improving ASU software development organization.

In the predesign stage, developers and customers face the problem of choosing a set of software quality indicators, prescribed and base values for these indicators, and methods of defining their qualitative values. The lack of generally accepted requirements and indicators describing the program product hinder formulating tasks (TZ) and specifications (TU), assessing decisions made, do not allow organizing industrial development of general-purpose programs, and lead to frequent changes in initial requirements and specifications, which in turn causes increases in system cost, extensions to development schedules and reductions in reliability. In a number of cases, more than half of all the problems occurring in implementing software systems are directly related to poor requirements definition.

The presence of a set of indicators for quality of programs is a necessary condition but insufficient for ensuring equivalence of compiled tasks and descriptions of tasks to those requirements posed by actual system operating conditions. Detailed, standardized, understandable and thoroughly prepared instructional materials on defining requirements, describing functions to be automated, structure and problem statements are needed. Also needed are

mathematical and informational models adequate for the classes of objects of control and controllable processes. Availability of such models would substantially facilitate choosing a general strategy for software development and standard data processing facilities, including data base management systems, previously developed.

To successfully solve these problems, objects and tasks to be automated have to be classified based on the commonality of their functional, structural and informational characteristics. Such classification will allow extensive development of standard software and use of it for various objects with minimal costs for adaptation to specific conditions.

Without classification of the objects to be controlled, developers and customers are forced to draft tasks and specifications and the description of problem statements (specifications) by orienting themselves to the specific object to be controlled. But any program may be proper only for a concrete specification. When specifications have been written for a specific object (system), the software developed in the best case will function properly under the specific conditions for this object.

For example, the ASU for the Krivorozhstal' [Krivoy Rog Steel] Plant was developed from the very start primarily for the conditions in this plant. Subsequently, the service programs in this ASU software were used in more than 30 enterprises. Naturally, there were some costs in each enterprise for adapting the software to the new specific conditions of use. These costs could have been reduced if the software developer had considered the specifics of a class of objects from the very start; the additional development costs could have been paid by the ministries and departments concerned.

The ASU system software for the Krivorozhstal' Plant was used even in enterprises where the production process would seem to have nothing in common with the production conditions for this plant (for example, the Minsk Mechanical Plant imeni Vavilov, the Khar'kov FED Association, the Bryansk IVTs [Information and Computing Center] for Trade Management, the Kiev Republic Information and Computing Center for the UkSSR Ministry of Health, and others). This indicates the presence of identity of tasks for information acquisition and processing in enterprises with dissimilar production. Identification, standardization and description of these tasks will allow centralized development, supply and maintenance of software, which in turn should substantially reduce costs for ASU introduction.

It should be noted that in the GosFAP [State Library of Algorithms and Programs] system, there arenow a considerable number of algorithms, programs and application software packages. But extensive use of them is hindered by the lack of simple, understandable, sufficiently universal methods and instructions on adapting these facilities to the specific conditions of enterprises (industries). At present, the developers of these facilities have no material interest in dissemination of them and bear no responsibility for their quality or maintenance.

Introducing an ASU and its software, just as any introduction of new technology, requires diverting at a certain time certain labor and physical resources of enterprises being automated from immediate fulfillment of current plan quotas and attaching them to the ASU development and introduction. These costs will be recovered only in the future. But in many cases, managers at various levels because of their focus on current problems and sometimes even certain conservatism underestimate the utility and objective necessity of timely allocation of certain resources to ASU development and introduction. But without the active participation of users in the ASU development process at the earliest stages, the project has less chance of succeeding. Getting into ASU software introduction in the later stages (out of necessity), users find numerous problems and mismatches which have to be resolved. But eliminating errors in a working system is ten times more expensive than if this had been done in the design stage.

Thus, it is necessary to increase the responsibility of managers in enterprises being automated to ensure timely commitment of hardware and solve other organizational-technical questions related to preparing objects for introduction and trial operation of an ASU.

Attention should especially be paid to the problem of ensuring software quality since ASU efficiency substantially depends on it.

The problem of attaining and maintaining the required level of ASU software quality, in our view, should be resolved in all phases of its life cycle. To this end, in enterprises (in sectors), developing and (or) operating software, within the bounds of complex systems for product quality control, software quality control subsystems with the following basic functions should be developed:
--forecasting fields of application of specific types of programs;
--planning quality of programs (establishing substantiated specifications for development of programs with the required values of quality indicators);
--checking design (checking the process of development of algorithms, programs and program documentation);
--testing (experimental determination of quantitative or qualitative characteristics of properties of programs);
--operational checking (checking and supporting the required level of quality of programs in the process of running them);
--maintenance (modification of programs, caused by necessity of eliminating bugs and (or) changes in functional characteristics).

Forecasting fields of application of programs should facilitate development of universal software suited to use under a broad range of conditions.

Quality planning consists in choosing a set of quality indicators, prescribed and base values of these indicators, drafting tasks (specifications) and functional specifications.

Design is checked at checkpoints which can be selected arbitrarily. But there are checkpoints which can be considered standard. These include drafting

tasks and approving them, drafting of technical proposals, drafting a common structural scheme, completing design of basic modules (according to a list) in the phase of draft design, etc.

In the most intense periods of efforts and in completing the most critical projects, it is advisable to require weekly reports from programmers on a specific form. These reports allow assessing the status of efforts at checkpoints and taking managerial actions when required.

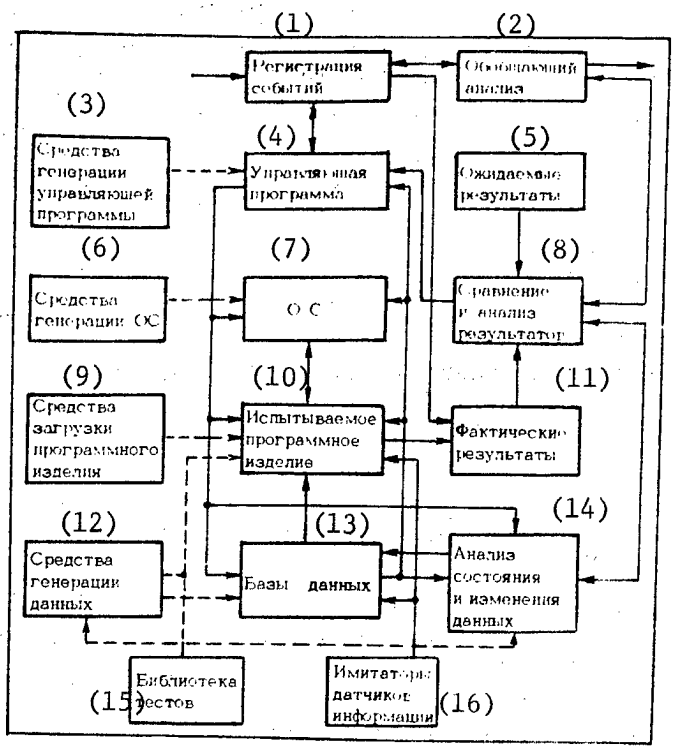
Program testing is a major element in the software quality management subsystem. Development of the most important programs should be completed not by checking serviceability in check examples, but by comprehensive tests. At scientific research institutes and design bureaus specializing in software development, it is advisable to set up software test sections (IPP) where software test specialists, and the appropriate test material, informational and software support should be concentrated

At the Kiev Institute of Automation, the problem of creating IPP's arose quite critically in connection with expanding the scope of efforts related to process control system development. To debug and test process control system software, several standalone test stands were developed; the needs for them for testing new systems are continually increasing. But each stand must have certain single-type modules (simulators of random effects, models of transducers of information and some physical processes, programs for statistical processing of information etc.). This circumstance led to the idea of developing a standard IPP. This IPP is a complex of hardware and software designed to test programs under conditions approximating operational. Successful completion of tests of major programs under test conditions should be a guarantee of their quality. Developing specialized IPP's to test programs in a specific class is advisable also from the economic view since this will prevent parallel development of single-type test facilities in various shops.

Also, modeling facilities concentrated in the IPP will allow performing intensive development of programs prior to completion of installation and checkout of system hardware in the object, which will substantially curtail the MIS development schedule. The IPP hardware base should include computers, facilities for data communication and teleprocessing, and process simulators.

The IPP software should include:
--information transducer simulators;
--programs for analysis and statistical processing of test results;
--control programs;
--check tests and programs for checking certain specific features of the programs being tested (analyzer of agreement in measurement units, determiner of assignment and calls, dynamic analyzer of branches, etc.);
--facilities for generating control programs, data bases and operating systems for specific use; and
--facilities for loading the program to be tested into computer memory.

The IPP information base is a specialized data base (bank).



Generalized test stand software and information support

Key:

1. event recording
2. generalizing analysis
3. control program generator
4. control program
5. expected results
6. OS generator
7. OS [operating system]
8. result comparison and analysis
9. software loader
10. program undergoing test
11. actual results
12. data generator
13. data bases
14. data status and change analysis
15. library of tests
16. information transducer simulators

All facilities for testing, modeling and statistical processing of test results should be developed as modules suited to repeated use and kept in the IPP library. To test specific programs, the IPP hardware and software are built into test stands: a set of hardware and software for generating test programs, simulating data and effects, recording information and maintaining control over the test process.

A version of a generalized structural scheme for program and information support for a test stand is shown in the diagram: The broken lines denote links between the IPP kernel and auxiliary programs used in preparing the stand for operation; program complexes, programs and modules in the stand kernel are linked by solid lines.

The tests, all types of checks and the software quality control process as a whole should be based on a particular system (set) of quality indicators. The basis of the system may be made up of these indicator groups:
--purpose (correspondence between its functional purpose and field of application);
--functional (suitability to direct conversion of source data to result sought);
--operational (fitness to maintaining serviceability, expeditious preparation and inclusion into operation, analysis of results of operation and modification);
--structural (informativeness, efficiency of links between components, integrity of composition and sophistication in programming style);
--feasibility (optimality of computer resource costs and labor costs for commissioning and operation under the specific conditions for a specific user);
--universality and standardization (adaptability to use under conditions different from those for which the software was directly developed, and saturation with standard, unified and original elements);
--economic (costs for making a prototype, duplication, generation, mastering, service and maintenance, and production costs for the program and documentation).

Analysts estimate labor intensiveness of software test and maintenance phases accounts for up to 80 percent of total development costs [1, 2]. This factor is, unfortunately, ignored in a number of cases. Software not really tested, but only checked with test examples is placed in trial (sometimes even in industrial) operation. This circumstance retards the process of introducing programs, and sometimes engenders distrust of them and computer data processing as a whole because significant number of bugs are found during operation.

Organizational-methodological support for a software quality control subsystem should be based on a complex of normative-technical and methodological documents prescribing the procedure for performance of efforts and defining the most efficient ways and means of reaching the ultimate goal of the subsystem: the required level of program quality.

A special role in this complex should be allocated to standardization, an efficient method of attaining a high level of quality. Needed primarily are state and sector standards prescribing the organization of efforts on program development, testing, introduction, operation and maintenance.

Conclusion. Thus, let us formulate some general principles for organizing ASU software development.

Customers and users should actively participate in ASU software development in the earliest stages (from the start of object surveys). In doing so, user responsibility for preparing the required hardware base and taking preparatory organizational measures should be increased.

Thorough drafting of technical tasks and functional specifications is a necessary condition for timely and qualitative completion of projects.

It is necessary to more extensively shift from development of individual ASU software projects to development of universal software suited to use in a hardware and software environment different from that for which the software was directly developed. Objects of control, tasks and subsystems should be classified by their identity features.

To limit users receiving poor quality software, it is necessary to subject it to thorough testing, and plan in advance for the appropriate labor, time and physical resources. In the lead scientific and research institutes and design bureaus, it is advisable to develop special test sections where hardware, software and specialists in testing programs will be concentrated.

For the program maintenance phase, it is necessary to plan for up to 60 percent of all labor costs for development and support of software serviceability. In the process, the technology of maintenance and level of responsibility of executors should be defined in advance.

To achieve and maintain high quality of software produced by different organizations, it is necessary to create a unified quality control system covering all phases in the product life cycle. This type of system should be supported by the apprpriate normative-technical documents, instructional materials and technical base.

BIBLIOGRAPHY

1. Glass, R., "Rukovodstvo po nadezhnomu programmirovaniyu" [Handbook on Reliable Programming, Moscow, Finansy i statistika, 1982, 256 pages.

2. (Zelkovits, M; Shou, A.; and Gannon, J.), "Principles of Software Development," Moscow, Mir, 1982, 368 pages.

8545
CSO: 1863/80

CENTRALIZED INTRODUCTION AND MAINTENANCE OF SOFTWARE THROUGH FUND OF ALGORITHMS AND PROGRAMS OF UKRAINIAN SSR ACADEMY OF SCIENCES

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 pp 115-116

[Article by S. A. Dorozhkin]

[Text] The State Fund of Algorithms and Programs (GosFAP) was organized in 1966 as a constituent part of the State Scientific and Technical Information System (GSNTI) to increase the efficiency of computer use in the national economy.

GosFAP was created on the territorial-sector principle and consists of central information, intersector specialized, sector (departmental) and republic (territorial) funds and algorithms of programs. Methodical supervision of the makeup and management of GosFAP is entrusted to the All-Union Scientific Research Institute of Problems of Organization and Management, while management of the central information fund is entrusted to the All-Union Scientific and Technical Information Center, GKNT [State Committee for Science and Technology] (VNTITsentr).

A departmental fund of algorithms and programs of the Ukrainian SSR Academy of Sciences (FAP AN URSSR) was created in 1978, prior to which the problem of providing institutions and organizations of the Ukrainian SSR Academy of Sciences with software for automation of scientific research, gathering data of the needs for software and use of it and introduction and maintenance of software was posed. The fund is managed by the SKTB [Special Design and Production Office] for software of the scientific and technical complex of the Institute of Cybernetics imeni V. M. Glushkov, Ukrainian SSR Academy of Sciences.

Programs of scientific and technical calculations, scientific data processing systems, information and teaching systems and also programs in the field of software engineering are deposited in the fund. There are individual programs, applications program packages and program complexes for second- and third-generation computers in the fund.

All software existing in the fund is divided into program and information sections. The program fund includes programs on machine carriers, composed of the necessary documentation according to the YeSPD [Unified System of Program Documentation] ready for immediate use on YeS, BESM-6, ASVT and other computers. The information fund consists of software for obsolete computers (Minsk-22, Minsk-32,

Promin' and so on) and software that has not been requested for a long time. Moreover, the information fund includes information materials that contain data on the software of the program fund, methodical, normative and instructive materials on working with software and on problems of formation, management and use of the fund.

The program fund contains more than 600 software programs, among which is software that enjoys increased demand: data base management system (Oka), program generation system for loading and realization of data bases organized by using the Oka SUBD [data base management system] (KOMPAKT), terminal polling system based on Oka SUBD (TOS), hierarchical data base linearization system (SLIBD), automated output data formating system (VYVOD), automated program production system (APROP), discrete programming applications program package (DISPRO), program complex for solution of systems of linear algebraic equations (PACTSIST-2), experimental data statistical processing applications program packets (Statistika), Prochnost'-75 system, training course programming system (SPOK), data telemanagement system (Kama), integrated multiprogram data processing system (SIMOD), distributed data base management system (Orbita), interactive query processing program for online testing of data bases (Interpoz), applications program packets for nondifferentiable and stochastic optimization (NDO), applications program package for automated synthesis of tests for digital units (Aist), program package for solution of plane and axisymmetrical problems of elasticity theory and thermal conductivity by finite element method on YeS computer (Element-1), information reference message processing system (Orakul'-II) and so on.

The compulsory program documents for software turned over to the fund should include program and maintenance documentation. The program documentation in turn includes:

specifications (GOST 19.202-78);

systems programmer manual (with test example) (GOST 19.503-79);

programmer manual (GOST 19.504-79);

operator manual (GOST 19.505-79);

program text (GOST 19.401-79);

application description (GOST 19.502-78);

program description (GOST 19.402-78).

The maintenance documentation includes a maintenance letter, information card, expertise report, review of specialist with indication of innovation and also a report on experimental testing or a copy of the introduction report.

The content and formulation of program documents and programs on machine carriers should meet the requirements of the state standards of the unified

program documentation system and should correspond to the "Instructions on the order of preparation and formulation of materials presented to GosFAP."

Lists of preliminary data on the software to be turned over are transmitted to organizations for preliminary data gathering on the software and for determination of the possibility of including them in the FAP. After the list has been received from the organization, the FAP is publicized for a month on the decision made on the problem of presenting the work for consideration. The presented software is made available in the program or information funds only after positive conclusions of the expert committee of FAP on the conformity of the given software to the topics of the fund and after preliminary evaluation of its innovation and demand.

All the software accepted in the FAP is sent to the VNTITsentr for recording. According to regulations on the procedure of awarding scientific degrees and scientific ranks, the algorithms and programs included in GosFAP are made equivalent to published papers.

The managing organization of FAP AN USSR offers the following main types of services:

delivery of copies of programs on machine carriers to user organizations with sets of operating documents without giving them the right to copy and transmit them to other organizations;

transmission of individual applications packages developed by the SKTB for software, the SKB [Special Design Office] of mathematical machines, IK AN URSSR [Institute of Cybernetics, Ukrainian SSR Academy of Sciences] and IK AN URSSR, to organizations provided an agreement is concluded on transfer of scientific and technical advances that provides for payment of bonuses to the developer, introduction of the software at the client's, and author maintenance;

transfer of duplicates of software to organizations who manage other FAP with the right to distribute copies of these programs with sets of operating documents among the enterprises and organizations of their own sector (department);

introduction and maintenance of software transferred to user organizations;

preparation and issue of recommendations and methodical materials to organizations of the Ukrainian SSR Academy of Sciences on the use of standard software;

conducting economic research and analysis of the economic effect due to introduction of software acquired by user organizations in the FAP AN USSR;

transfer of copies of lists to developer organizations on the economic effect achieved due to introduction of the software developed by them;

reference information servicing of organizations.

The FAP implements reference and information servicing of user organizations provided an agreement is concluded, according to which the following types of services are offered:

servicing according to the selective information distribution system (signal cards with information about new arrivals of software in the FAP are sent according to the topical queries of organizations);

distribution of "Annotated indices of materials on applications software for computers" (two issues annually);

distribution of a catalog of programs and an annual supplement to it;

provision of bibliographic information on a one-time query;

methodical assistance in organization of the work of the scientific and technical information service.

Services on transfer of software and of reference and information servicing for academic institutions of the Ukrainian SSR Academy of Sciences are performed free of charge. The enumerated services are performed for similar organizations according to written applications and after payment of their account. The services offered by user organizations of FAP AN USSR are paid for at established and confirmed prices and rates or according to estimated calculation confirmed by the director of the organization managing the FAP AN USSR.

Written applications to acquire programs and reference and information servicing should be sent to the director of SKTB PO [software] to the address: 252207, Kiev, Prospekt Akademika Glushkova, 20.

Zapadnyy and Donetsk branches of FAP AN USSR have been created to increase the efficiency of using computer technology at institutions and organizations of the scientific centers of the Ukrainian SSR Academy of Sciences. The Donetsk branch functions as part of the SKTB of management systems, Institute of Applied Mathematics and Mechanics, Ukrainian SSR Academy of Sciences (340100, Donetsk, Teatral'nyy pr., 23, telephone 90-83-21), while the Zapadnyy branch functions as part of the computer center, Institute of Applied Problems of Mechanics and Mathematics, Ukrainian SSR Academy of Sciencies (290005, Lvov, ul. Lermontova, 15, telephone 72-40-86).

6521
CSO: 1863/122

UDC 680.3

INVESTIGATION OF COMPUTING PROCESS IN YES OPERATING SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84
(manuscript received 8 Jul 83) pp 42-47

[Article by V. A. Korotkevich, I. V. Maksimey and A. B. Demus'kov]

[Excerpts] Conditions for conducting investigations. The job batch processing mode in the YeS operating system was investigated in the STATOS program package [1] under conditions of the VTs [Computer Center], SO [Siberian Department], AN SSSR [USSR Academy of Sciences].

The investigation was conducted on the YeS 1052 computer with main memory capacity of 1 Mbyte. The peripheral equipment of the computer included eight YeS 5061 disk units with access to two selector channels through a single YeS 5561 controller. The investigation was conducted in a real industrial operating situation of the YeS operating system in the MVT mode, which functions under the control of a shared-resource dispatcher [2], with multiprogramming factor for user tasks of approximately 3 (three logic sections of the KROS scheduling subsystem were used).

The main purpose of the investigation was to construct semi-Markov models of the computing process in the YeS computer, which can subsequently be used in design of computer systems based on YeS computers. These models, designed by using real statistics, may be useful to designers in the following cases: in development of multimachine computer systems based on YeS computers and a set of remote intelligent terminals, in working out the principles of the computing process in a VTsKP [Collective Use Computer Center] [3] and in design of special programs intended for operation in a given mixture of tasks and being limited to execution time. Two other goals were also pursued: to estimate the structure of the overhead expenses of the YeS operating system on organization of the computing process in a real task flow (expenditures of systems programs and of the supervisor and down times of the processor), to direct attention of the developers of operating systems to the most actively used components of the supervisor, which is important if modification of the supervisor of the YeS operating system is planned. Three task packages were configured arbitrarily for the investigation. The statistics of realization of the computing process using the STATIST program, included in the STATOS package, was reported during each session of solution of one of the test packages. The length of the measurement sessions comprised 120, 110 and 250

minutes, respectively, for the three packages. All three packages mainly contained user debugging tasks, which were solved in the batch mode under conditions of the Computer Center, Siberian Department, USSR Academy of Sciences. The results of recording the statistics were stored in the form of three sets of data, each of which was processed separately by the STATPRC program, included in the STATOS package and that utilizes the procedure of SPRINT statistical analysis program package.

The results of processing were combined for all sessions. A characteristic feature of the investigation was low timer resolution of the YeS 1052 computer, which is approximately 20 ms, as a result of which many time characteristics were measured with wide dispersion and the mode of the distribution histograms of these characteristics is in the region of the first interval of values.

Conclusions. As a result of the investigations on the YeS computer, three types of semi-Markov models were obtained, joint use of which provides simulation of the computing process in the YeS operating system:

a functional model of user tasks that permits time representation of requests for different types of service to the supervisor of the YeS operating system;

a functional model of systems tasks that permits one to simulate the requests of these tasks to the supervisor of the YeS operating system;

a functional model of the supervisor of the YeS operating system that permits one to simulate the servicing of requests during multiprogram processing of tasks.

The composition of the overhead expenses of the YeS operating system on organization of the computing process was evaluated. The poor load of the central processor during operation of the system was determined, which indicates nonconformity between the processor speed, main memory capacity and number of independent access paths to the disk units in the YeS 1052 computer used at the Computer Center, Siberian Department, USSR Academy of Sciences.

The most actively used supervisor components of the YeS operating system were determined, which include: an input-output supervisor, main memory supervisor and task synchronization devices.

6521
CSO: 1863/122

UDC 681.3.06

APPROACH TO DEVELOPING REAL-TIME OPERATING SYSTEMS FOR PROBLEM-ORIENTED COMPUTER COMPLEXES

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 15 Sep 82) pp 61-65

[Article by Aleksandr Nikolayevich Dolgov, engineer, Krasnyy Hydraulic Press Plant (Taganrog)]

[Text] The emergence of new generations of microcomputers with advanced architecture (like the Elektronika-60, the Elektronika-NTs and the SM-300) is creating the prerequisites for development of on-board multimachine and multiprocessor computer complexes (MVK), including problem-oriented, based on series microcomputers and special processors. A microprocessor system application is automating information processing in fish-hunting systems [1]. The large amount of data obtained from sonar, navigational and other instruments must be processed in real time; the results of the fishing situation must be displayed in graphic and alphanumeric form; this requires, as foreign experience shows, using multiprocessor, problem-oriented, computer complexes (POK).

The class of multiprocessor computer complexes in question is a decentralized computer structure with asynchronous organization of the computing process in real time. This structure and organization of the computing process stems from the following problem orientation and functional purpose of the complex:
--entry and preliminary processing of information and output of it to alphanumeric and graphic displays;
--output of processed information and control signals to peripherals in the automated system for finding fish.

Automated systems for finding fish (ARPK), based on multiprocessor computer systems, may include an operator communication unit allowing entry into the computer system (VS) of operational information for tuning it to the composition and parameters of information sensors, and data describing the external environment. In some cases, to speed up processing of sonar data, it is advisable to include special processors in the multiprocessor computer system.

One of the main questions occurring in organizing the computing process in a multiprocessor computer complex is development of a real-time OS. Although the problems in development of real-time system software have been covered in a rather large number of publications, most fundamental of which are [2-4],

the relevance of this problem and interest in it in connection with extensive use of microprocessor technology are constantly growing.

This article is devoted to describing an approach to development of a real-time OS for problem-oriented computer complexes based on series microcomputers, the architectural features of which are similar to those of the SM-3 and SM-4 minicomputers.

Since a dual-machine, specified, control computer complex based on the SM-4 control computer complex was used as the computer system for the first prototypes of automated systems for finding fish, it was logical to analyze the functional capabilities and characteristics of standard real-time OS's for minicomputers in this class [5,6]. But all these real-time OS's, just as the facilities offered in their generation, do not meet the basic requirements for operating systems in such computer systems:
--Memory taken up by the real-time OS for each of the minicomputers and microcomputers must not exceed 2K words;
--There must be service, with strict time limits, of a specific set of peripherals (unit for I/O of analog and discrete information, graphics display, special processors and others);
--Diagnostic checking of each minicomputer and microcomputer and the problem-oriented computer complex as a whole must be provided.

Also, the following requirements are imposed on the software (PO) for this type of computer system:
--all software must be resident (for microprocessor systems, software must be in ROM);
--systems software (operating system nucleus) must have minimal functional redundancy (i.e. be problem-oriented), high efficiency, and properties of unification and adaptation to ensure the capability of adding on to the computer system and modifying it.

Considering all this, the problem of developing a specialized real-time OS for the problem-oriented computer complex was posed. This real-time OS may be classed as executive [7].

The main functions of the real-time OS for each microcomputer in the problem-oriented computer complex may be [8-12]: allocation of processor time, time service, organization of information I/O, synchronization of tasks, and diagnostics.

Advanced in [13] is the idea that operating systems for mini and micro computers in their current form will probably disappear in future; replacing them will be various components of an operating system, including hardware implementations [14] (this is most probable and relevant to real-time OS's in microprocessor computer systems), with regard to recommendations on software structure development for real-time ASU [automated control systems] [10]. This will allow developing a trilevel hierarchical decomposition of the software structure (fig. 1) with representation of the real-time OS for the problem-oriented computer complex as a set of universal operating modules.

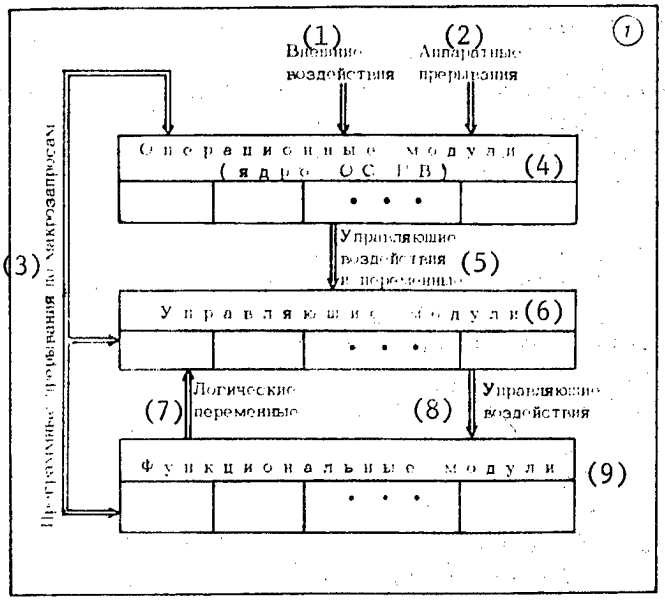


Fig. 1. Trilevel hierarchical software structure

Key:

1. external actions
2. hardware interrupts
3. software interrupts for macro-requests
4. operating modules (real-time OS nucleus)
5. control actions and variables
6. control modules
7. logic variables
8. control actions

The first level of the hierarchical software structure is a set of operating modules, the real-time OS nucleus. The planning strategy is not included in the real-time OS nucleus, similar to [11]. It is defined by the second level of the hierachical software structure, the control modules. The operating modules are the program interface between the hardware and lower levels of the software structure. For example, one can cite the following list of operating modules enabling efficient functioning of the problem-oriented computer complex in real time: initialization and adaptation, information I/O, one for keeping track of system time and scheduling the start of a task upon expiration of specified time intervals, task synchronization, priority control, operator communication, one for communication with the specialized processor and the diagnostics module.

For a problem-oriented computer complex with a centralized computer structure or with functional redundancy, operating modules implementing the functions of inter-machine communication can also be developed [8].

By functional purpose, the control modules are dispatchers of priority levels [15]; they are not universal and are oriented to supporting a particular

computer process for each microcomputer in the problem-oriented computer complex. Joint use of operating and control modules allows synthesizing any strategy for servicing requests in the system.

The third level in the hierarchical software structure contains functional modules enabling implementation of the basic functions in the automated control system being developed.

In the suggested hierachical software structure, external actions enter the system through the operating modules (top down); control information enters as macro-requests from the control and functional modules (bottom up).

Based on the main concepts of operating systems [16], the following principles have been formulated and implemented for developing the real-time OS nucleus for the problem-oriented computer complex:

Modular Structure. The operating system nucleus is divided into separate, independent operating modules. An operating module implements one of the operating system functions and receives control by hardware interrupts. The modular structure of the real-time OS nucleus allows assembling individual microcomputers in the problem-oriented computer complex by various combinations of operating modules oriented to handling specific functional tasks in real time. This real-time OS nucleus structure allows replacing individual modules, modifying them and adding new ones in the system development process.

For the majority of mini and micro computer real-time OS's [5, 6, 11], system nucleus modules are linked together by common information tables, task description blocks, etc., which restricts the flexibility of the operating system nucleus. Also, standard operating systems are functionally complete, having a rather large degree of universality. The suggested operating modules making up the real-time OS nucleus feature independence of each other.

Parametric Universality. Since the microcomputers making up the problem-oriented computer complex can be fitted with various peripherals and have various characteristics of similar peripherals, parametric universality is required of the operating modules. Operating modules are configured to the required parameters during system generation.

Functional Selectivity. This principle consists in outfitting each microcomputer in the problem-oriented computer complex with the needed number of operating modules as a function of characteristics of tasks to be handled and microcomputer configuration.

Principle of Generatability. This principle is implemented by using a program generator, the source information of which are operator directives describing the problem-oriented computer complex configuration and allocation of functional tasks in the complex. In the system generation process, the operating, control and functional modules are automatically assembled into a single software system for each microcomputer separately (fig. 2).

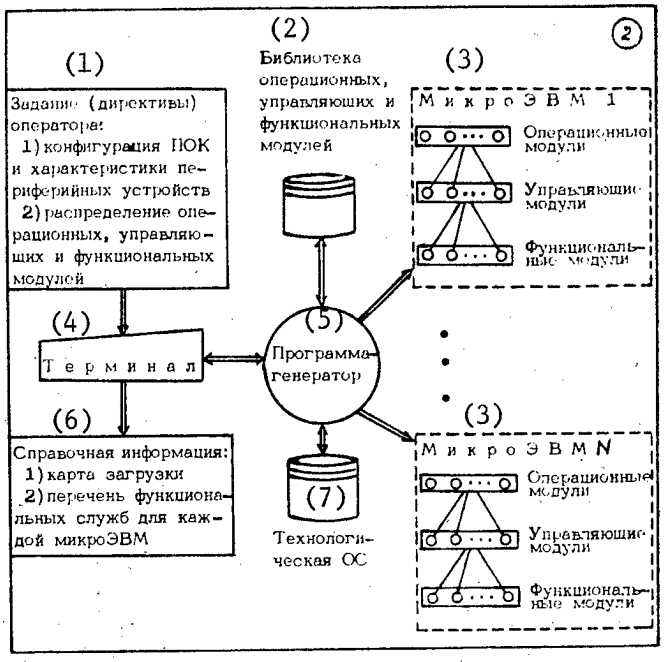


Fig. 2. System generation process

Key:

1. operator's job (directives):
   1) configuration of problem-oriented computer complex and characteristics of peripherals
   2) allocation of operating, control and functional modules
2. library of operating, control and functional modules
3. microcomputer 1, N: operating, control and functional modules
4. terminal
5. program generator
6. reference information:
   1) load map
   2) list of functional services for each microcomputer
7. process OS

By operator directives in the generation process, only the modules needed are assembled. Implementing the operating modules with regard to the principle of parametric universality allows automatic configuration of them for the parameters specified during system generation (for example, specifying the maximum possible number of tasks to be started upon expiration of specified time intervals, specifying addresses, types and characteristics of peripherals).

At the end of system generation, the operator can request reference data on the system generated (maps for loading of programs, list of functional services of the operating modules for each microcomputer in the problem-oriented computer complex, etc.). In contrast to the procedure for generating an executive OS for a single-processor computer [19], this procedure allows

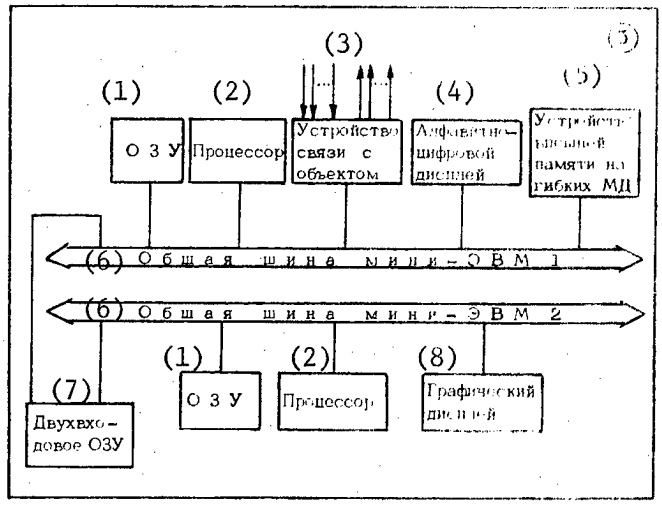


Fig. 3. Specified control computer complex

Key:

1. main memory
2. processor
3. unit for communication with object
4. alphanumeric display
5. floppy disk drive
6. common bus, minicomputer 1, 2
7. dual-port main memory
8. graphics display

developing assembled software (operating, control and functional modules) for multiprocessor computer systems.

Translation, offline debugging and generation are performed with a production OS. In doing so, a microcomputer is used with expanded configuration which has to include magnetic media storage units, on which the operating modules are kept in appropriate libraries. After all types of complex debugging and comprehensive tests of the problem-oriented computer complex making up the automated control system under development, the software can be placed in ROM and the system will be ready for operation.

This approach to developing a real-time OS was used in developing an automated system for finding fish, used to display the fishing situation when using a seine net. The computer system in this complex was implemented with a specified control computer complex based on two SM-4 control computer complexes joined by a common block of main memory (fig. 3). The object communication unit (USO) is used to communicate with the sonar and navigational instruments.

The EPG-SM graphics display is used to display the fishing situation. The specified control computer complex software is stored on floppy disks.

When the complex is turned on, a special loader loads the software from floppy disks into both minicomputers and it then becomes resident. Memory taken up by the operating modules for one of the minicomputers was about 1.5K words; for the other, about 1K words (because of the different number of operating modules included during generation). For the problem-oriented computer complex based on the SM-4 control computer complexes, the following operating modules were developed: initialization, I/O of analog and discrete information, handling of interrupts from external initiative signals, keeping track of system time, planning of start of tasks upon expiration of specified time intervals, system diagnostics and operator communication.

The system initialization module clears all peripheral status registers, sets up a stack, sets processor priority to zero, and sets up all interrupt vectors and readiness of I/O units. During system generation, the initialization module is configured for the number, address and types of peripherals.

The modules for I/O of analog and discrete information are included in the software for the minicomputers specified during generation and by operator directives are configured for the addresses to which the corresponding devices are connected. The information I/O modules are accessed from the functional modules in accordance with agreements on communications adopted in the system.

For modules handling interrupts from external initiative signals, a similar software configuration procedure is used during system generation plus the type of incoming signal (lock/unlock) is considered. The interrupt handling modules transfer control to the appropriate control modules and reset the readiness bit for a given device for this time.

The system time module stores the time from the timer and handles macro-requests for determining system time.

The module for planning the start of tasks upon expiration of specified time intervals handles macro-requests for starting tasks at the designated time and those to cancel them. During system generation, operator directives specify the maximal number of tasks to be started by time for each minicomputer.

The module for communication with the operator implements macro-requests for output of character strings to the alphanumeric video terminal and entry of information from its keyboard upon request.

The system diagnostic module handles malfunctions occurring on hardware interrupts, analyzes them and codes the error. During system generation, the module is configured for possible types of interrupts and receives the identifier of the specified minicomputer. During error handling, coded information is placed in the common memory field and then decoded and output to the display.

Here are the characteristics for the set of operating modules developed.

Technical characteristics

Memory occupied by operating modules	not over 1.5K words
Program operating time:	
entry of discrete signals	not over 10 microseconds
output of discrete signals	not over 50 microseconds
entry of initiative signals	not over 20 microseconds
entry of analog signals	
a) without switching of channels for analog-to-digital conversion (ADC)	not over 30 microseconds
b) with switching of ADC channels	not over 80 microseconds
output of analog signals	not over 25 microseconds
Memory occupied by all I/O programs	not over 150 words
Time of operation of module for starting tasks upon expiration of specified time intervals	not over 400 microseconds

From this data, the conclusion can be drawn that the set has the following basic advantages compared to one of the most responsive real-time OS's, the FOBOS for the SM-4 control computer complex:
--Main memory occupied and time of operation of I/O modules for the set in question is five-fold less;
--Rate of handling requests by the module for starting tasks upon expiration of specified time intervals is more than 1.5-fold more efficient than the same FOBOS function;
--Main memory occupied by the set of operating modules is less than the minimum possible size of the FOBOS resident portion.

These advantages for the suggested set of operating modules were achieved because they have no redundant functional capabilities such as management of memory, protection of memory, file management, control of operator and modes of system operation, etc. that are typical of real-time production OS's.

The FOBOS operating system was used as the production OS for software translation and debugging. For system software generation, the SYSMAC. SML library was supplemented with macro-definitions linking the functional modules to the operating modules. A special program is used for the generation procedure. The generation program operates under control of the FOBOS operating system in the interactive mode and writes the assembled software (functional, control and operating modules) to a floppy disk as separate files for each minicomputer. During operation, the generation program uses these system programs in the FOBOS operating system: PIP for operation with files, MACRO compiler with macro-assembler, LINK, and the BATCH processing program.

The main advantage of this approach to developing executive real-time OS's for problem-oriented computer complexes is the capability of modular minimization of the real-time OS nucleus and its high efficiency. Using the operating modules in combination with the trilevel hierarchical structure of the system

software not only increases system flexibility and facilitates adding and modifying modules in the different levels in the hierarchy, but also simplifies phased debugging of the system as a whole.

This approach is especially effective for real-time automated control systems in which the need for reconfiguration of computer complexes or changing the organization of the computing process arises rather often, and for systems having a large number of modifications.

BIBLIOGRAPHY

1. Tesler, V. D., "Prospects for Using Computers on Industrial and Scientific Research Vessels," PROMYSLOVAYA RADIOELEKTRONNAYA APPARATURA I PODVODNAYA TEKHNIKA, No 2, 1981, pp 1-43.

2. Lipayev, V. V.; Kolin, K. K.; and Serebrovskiy, L. A., "Matematicheskoye obespecheniye upravlyayushchikh TsVM" [Digital Control Computer Software], Moscow, Sov. radio, 1972, 528 pages.

3. Martin, J., "Programming for Real-Time Systems," Moscow, Nauka, 1975, 360 pages.

4. Nikitin, A. I., "Obshcheye programmnoye obespecheniye sistem real'nogo vremeni" [General Real-Time System Software], Kiev, Nauk. dumka, 1980, 136 pages.

5. Rostokin, B. M., "Real-Time Operating Systems for the SM-3 and SM-4 Control Computer Complexes," PRIBORY I SISTEMY UPRAVLENIYA, No 12, 1977, pp 1-2.

6. Nemirovskiy, M. B.; Piyadin, A. P.; Rytvinskaya, M. S.; and Shtil'man, L. F., "Functional Capabilities of Real-Time Operating System (Real-time OS for SM-3 and SM-4 Control Computer Complexes)," TRUDY INEUM, No 86, 1981, First Phase Small Computer System Hardware and Software, pp 8-16.

7. Kibitkin, V. V., "Features of Microcomputer Operating Systems," USiM, No 1, 1982, pp 31-35.

8. Nikitin, A. I., "Functional Capabilities of Real-Time Operating Systems," USiM, No 5, 1973, pp 26-35.

9. (Kan, K. S.), "Principles of Designing a Small operating system for Microprocessors," TIIER, Vol 66, 1978, pp 125-143.

10. Lipayev, V. V., "Proyektirovaniye matematicheskogo obespecheniya ASU (sistemotekhnika, arkhitektura, tekhnologiya) [Automated Control System Software Design (Systems Engineering, Architecture, Technology), Moscow, Sov. radio, 1977, 400 pages.

11. Mytus, L. L., "Real-Time Operating Systems Used in Process Control Systems (Review of Foreign Systems), PROGRAMMIROVANIYE, No 6, 1977, pp 44-56.

12. Burton, D. P. and Dexter, A. L., "Handle Microcomputer I/O Efficiently," ELECTRONIC DESIGN, No 13, 1978, pp 70-78.

13. (Pike, H. E.), "Minicomputer Software Development," in "Minicomputers," Moscow, Mir, 1975, pp 27-41.

14. (Tunnon, J.), "Silicon Implementation of Executive Program in Real-Time Operating System," ELECTRONICS, No 8, 1982, pp 61-66 [Russian translation].

15. Ignat'yev, V. O.; Alekseyev, B. Ye.; and Rostikov, S. S., "Programmnoye obespecheniye ATS" [Automatic Telephone Station Software], Moscow, Radio i svyaz', 1981, 176 pages.

16. Solov'yev, G. N. and Nikitin, V. D., "Operatsionnyye sistemy tsifrovykh vychislitel'nykh mashin" [Digital Computer Operating Systems], Moscow, Mashinostroyeniye, 1977, 112 pages.

17. Johnson, R. C., "Operating Systems with Full Set of Facilities for Operation with 16-bit Microprocessors," ELECTRONICS, Vol 55, No 6, 1982, pp 33-45 [Russian translation]

18. Pohjanpalo, H., "MROS-68K: A Memory Resident Operating System for MC68000," SOFTWARE, Vol 11, No 8, 1981, pp 845-852.

19. Rub, W. and Schrott, G, "Automatic Generation of Operating System for Problem-Oriented Process Control Computer," ANGEWANDTE INFORMATIK, No 1, 1980, pp 7-16.

8545
CSO: 1863/80

UDC 681.3.06

UNIFIED OPERATING SYSTEM FOR ELEKTRONIKA-60, SM-3, SM-4 COMPUTERS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
(manuscript received 21 Jul 82) pp 65-69

[Article by Vladimir Vladimirovich Kibitkin, candidate of engineering sciences, USSR Academy of Sciences Scientific and Technical Association (Leningrad)]

[Text] Introduction. The concept of a family of computers, now generally accepted, was first used in designing general-purpose computer families such as the Unified System of Computers and the IBM/360. The widespread use of these computers has proven the high efficiency and practical value of this concept; it has begun to be used also in mini and micro computer development. A chief advantage of computer families is the capability of transferring applications software (with some limitations) between the various models in a family. This capability is based on software compatibility of the models, which from the view of applications software must include compatibility at the operating system (OS) level too. For general-purpose computers, this compatibility is achieved with no special complications; but achieving such compatibility for mini and micro computer families involves many difficulties. This work is aimed at examining these difficulties and ways of eliminating them for the Elektronika-60, SM-3, and SM-4 computers.

Specifics of Mini- and Micro-Computer OS. The specifics are defined by the features of using computers in this class. The first feature is that the overwhelming majority of these computers are used in real-time control computer complexes and handle the same fixed set of programs in the process. But cyclical implementation of the same set of programs during system operation requires less hardware resources and considerably less degree of dynamics of their allocation than implementation of a variable set of programs coming from an external source. Therefore, when these computers are operated in a control loop, the OS is simple and requires few computer resources to implement it.

Another feature of these computers is related to the capability of rather simple expansion of them within the bounds of the same architecture for execution of not only fixed, but also a variable set of programs. In practice, this capability leads to two new fields of application of these computers. The first is their use for debugging programs which after development will be implemented on the same minis or micros, but with a reduced configuration. The OS's

required in this case must include all the facilities for automating program development and must support the capability of multiuser debugging. The second is the use of these computers for both developing new programs and implementing those already developed; and in a number of cases, parallel execution (in the sense of multiprogramming) of the processes of program development and implementation is desirable. An OS combining the features listed above is required for this.

Thus, the presence of these fields of application for minis and micros is causing the emergence of three OS classes [1]:
--production, used for program development and debugging;
--executive, used to execute programs developed in the production OS environment;
--general functional, with functional properties of both production and executive OS's.

In using production and executive OS's, special attention should be paid to software compatibility between the various models in the mini and micro family at the OS level. As is known, any OS is in two forms: distributive and working. The simplest solution to the problem of software compatibility would be to use the same working OS for any model in the family with any form of its use. But in practice, this solution is inefficient (if possible at all) since the unique working OS will either have quite a lot of redundancy as an executive OS or low efficiency as a production or general functional OS. So for the executive, production and general functional OS's, different working versions of the OS, generated from the distributive form of a certain base OS common to the entire family of minis and micros, should be used. Let us look at the problems in choosing this common base OS for the Elektronika-60, SM-3 and SM-4 computer family.

There are now more than 10 different OS's for computers in the Small System family; so to select a base OS, a set of criteria must first be defined. It is proposed that the three fundamental criteria should be:
--the capability of implementing working OS's generated from the base on any processor in the Small System family;
--the capability of implementing working OS's generated from the base on both computers configured with up to 64K and those with up to 256K bytes of memory;
--the capability of placing the working OS and application programs on any of three media: floppy or hard disks or main memory (when there are no disks in the selected computer configuration).

These criteria are met by the RAFOS and the real-time OS. But RAFOS has a serious shortcoming compared to real-time OS: It essentially does not support multiprogramming, which is needed in the majority of cases for both control system software development and implementation. So let us look at the possibility of using the real-time OS as the basis of a common OS for the Small System computer family. A prerequisite for this is combining the capabilities of both production and executive OS's in the real-time OS.

Production OS's Based on the Real-Time OS. As noted earlier, production OS design must be based on the idea of implementing multiuser development and debugging of applications software, based on using a powerful, multi-terminal minicomputer (in our case, the SM-4, but in future, the SM-1420) which offers development and debugging at the same time (in the sense of multiprogramming) to several programmers. The multiuser mode of program development is required both from economic considerations and for ensuring the capability of joint use of common programs and data by several programmers. If each user had to debug his own portion of a general software system on a separate machine, there would be greater difficulties in maintaining all copies of common programs and data on the same level of updating.

A production OS consists of a control program, production facilities for programming (translators, text editors, linkage editors, debuggers, etc.) and various service routines (librarians, programs for copying data from one medium to another, printing data, etc.). The main feature of the control program in a production OS is the necessity of controlling execution of a variable set of programs since during debugging of programs, an instrumental minicomputer is used by various programmers who are continually modifying their programs. Thus, requests for minicomputer resources vary sharply.

Because of this, a chief requirement for the control program in a production OS is support of full automation of allocation of all types of resources and support of dynamics in resource allocation that is considerably higher in degree than in an executive OS.

The possibility of using the real-time OS as the basis for designing a highly efficient production OS is governed by these real-time OS functional capabilities:
--dynamic allocation of main memory in the system-control partition;
--support of multiuser mode of debugging programs based on facilities of recording, identifying and protecting users from each other, sophisticated file system, facilities for joint use of files, facilities for processor time slicing (by using circular dispatching);
--support of independence of programs from peripherals used;
--support of a sophisticated "user-OS" language interface based on the MCR program instructions, mechanism of indirect instruction files, instructions and keys of system programs used during debugging;
--high-level facilities for interactive development of programs;
--high-level facilities for automating programming and debugging;
--capability of obtaining, on the basis of real-time OS, various executive OS's that are software compatible with the production OS used;
--high-level facilities for controlling multimachine complexes based on the System of Small Computers.

Production OS's built with the real-time OS can be implemented on both the SM-3 and SM-4. When a large number of programmers is available, setting up a multiuser computing center for debugging software for various control systems which use the Small Computer family of computers is very promising. This computer center should have a complex of two-three SM-4 computers, with several

terminals connected to each of them; these machines should operate with a common extent of external storage. Such a center has special value to developers of control systems based on the Elektronika-60 computer.

Executive OS's Based on Real-Time OS. An executive OS is actually a set of control programs and a loader used to load finished programs. Since the set of finished programs implemented in the executive OS environment is known in advance, there is the capability of precisely defining the requirements for mini and micro resources. So in operating a computer within a control system, many resources can be shared in the applications software development phase; the degree of dynamics of allocation of resources which has to be supported by the executive OS is not large. In the extreme, but common, case, an executive OS must execute just four functions:
--allocation of processor time based on applications software priorities;
--execution of requests from applications software for sending messages among the programs;
--execution of requests from applications software for synchronizing the programs with each other;
--control of functioning of peripherals during operations for exchanging data between these devices and main memory.

The real-time OS supports these functions because it has:
--a mechanism for partitions controlled by the user;
--the DISPETCHER program which tracks the occurrence of so-called major events, to each of which program redispatching is related;
--language interface facilities supporting execution of a directive issued by applications software for synchronization of these programs and sending messages among them;
--a set of drivers and a file system.

In practice, the process of deriving an executive OS based on real-time OS depends to a very large extent on types of processor and external storage units. When an executive OS is to be implemented on an SM-3 or SM-4 computer with cartridge or floppy disks, the process of designing it is reduced to the usual procedure for generating the required system from the real-time OS distributive. But when the Elektronika-60 is used for implementing the executive OS, some expansion of real-time OS is required.

Let us first consider the non-disk Elektronika-60 configuration. In this case, both the applications software and the OS (which we will call resident for this case) must continually reside in main memory. In using an executive OS, a chief requirement is minimizing main memory.

Practice shows at least 5K bytes are needed for the real-time OS control program. With that, real-time OS capabilities are implemented only partially, but all basic functions are kept. To obtain a usable resident executive OS with minimal capabilities, drivers have to be added to the minimal control program. Two-four drivers requiring 3K bytes are needed to support operation with type configurations of Elektronika-60 peripherals (with no disks). Thus, the total size of a minimal resident executive OS is 8K bytes.

Resident executive OS capabilities can be expanded primarily by including new program facilities during OS generation. With that, the memory needed for the control program increases to 16K bytes.

Another way to expand resident executive OS capabilities is to include the operator communication routine (MCR) and file control system (FCS) in addition to the control program and drivers. But direct use in the resident executive OS of the MCR and FCS routines that are part of the real-time OS is not possible since they are extremely large and rely on disk storage in their operation. So these programs have to be revised to reduce their size. This revision is facilitated by the limitedness of the computer configuration in question since one can narrow the range of functions executed by these programs. Thus, instructions for mounting and removing magnetic disks, loading and set up of tasks, outputting information on devices, establishing the correspondence between logical and physical devices, etc. should be removed from the MCR. Thanks to this, MCR size can be reduced to 4K bytes and it can be placed entirely in main memory. As for the FCS, the main part of it consists of subroutines for maintaining the file structure on magnetic disks; removing these subroutines allows obtaining an FCS with 2.5K bytes.

A practical difficulty that occurs when including the operator communication routine and file control system in a resident executive OS is that these programs, in the form indicated above, cannot be directly derived from the MCR and FCS programs in the real-time OS generation process. So to obtain a resident executive OS based on real-time OS, separate versions of the programs in question, MCR subset and special FCS, which meet the requirements listed above, should be developed and included in the real-time OS distributives.

A resident executive OS can be derived for both the Elektronika-60 and the SM-3 and SM-4 computers. Because these machines have some architectural differences, computer type and presence or lack of a memory dispatcher should be indicated during generation of the resident executive OS.

Of essential importance is the question of linking applications software to a resident executive OS. The simplest solution is to link the applications software to it in the last phase of its generation by using the virtual operator communication routine (VMR). After this, the executive OS and applications software become a common software system which is loaded into the computer. But in many cases, the set of applications software executed under control of the same working version of the resident executive OS may have to be replaced periodically. So the OS should include a special program, the online loader (OTL), which can load new applications software in place of old into a computer in accordance with operator instructions; it needs 5K bytes for its operation. In practice, in a number of cases, the entire set of a resident executive OS and applications software, stored in computer main memory, may have to be backed up periodically on some external medium for subsequent reloading into main memory. For this, the resident executive OS should include the software system preservation program (SIP). The amount of memory needed for SIP depends on the diversity of the nomenclature of the external storage devices that can be used for software backup. When just one device is used, 3K bytes are needed for SIP operation.

Thus, a resident executive OS affording execution of applications software developed in the real-time OS environment can be derived when the four additional programs (MCR subset, special FCS, OTL and SIP) are included in the real-time OS distributives. This OS can fit entirely in main memory in computers in the Small System family and supports maximum spped of system operation. As a function of capabilities required, the executive OS needs from 8 to 25K bytes of main memory. Considering the Elektronika-60 and SM-3 have up to 64K while the SM-4 has up to 256K, this OS should be considered essentially efficient. This conclusion is confirmed also by the extensive use abroad of the RSX-11S OS [2] for the LSI-11 microcomputer and PDP-11 minicomputer. As a result of analysis of it, this method of deriving an executive OS based on real-time OS is also suggested.

Let us look at another configuration of the Elektronika-60 that is produced in series, one with floppy disks. The resident executive OS considered above can be used in this configuration; when the system is generated, it can include drivers for operation with floppy disk units. But including the drivers means only the capability of data exchange with floppies and no more. At the same time availability of floppies allows supporting dynamic pumping of program modules into RAM and organization of a file structure. Therefore, the resident executive OS has to be expanded.

Expanding a resident executive OS to support a file structure is inadvisable. The fact is that the full version of the FCS that is part of real-time OS is extremely large and cannot fit entirely in RAM. And the FCS cannot be placed on floppies because the resident executive OS derived from real-time OS does not permit use of separate parts of it as transits (i.e. kept on external media and called into RAM when needed). In practice, the lack of a file structure for the case in question is not very important since, when necessary, data kept on magnetic disks and processed in control systems based on the Elektronika-60 can rather simply be structured in applications software.

In the non-disk configuration of the Elektronika-60, all applications software must be loaded into RAM before execution; thus, the size of these programs is governed by the RAM size, which is a very critical limitation. Naturally, when floppies are available, it is extremely desirable to eliminate this limit. The common way to do this is to place all programs on disks and dynamically pump them into RAM. Obviously, the pump methods must be identical in the production and executive OS's.

The real-time OS supports three methods of dynamic pumping: organization of overlay structures, use of mechanism of task unloadability, and dynamic loading and execution of programs by using the RUN command. The resident executive OS derived from real-time OS does not support dynamic pumping. Including the first two methods of dynamic pumping, which are available in real-time OS, in the resident executive OS requires substantial revision of the latter, but the third method can be implemented by expanding OTL. The expansion consists in the fact that OTL executes its functions not only be operator instructions, but also by commands from applications software, i.e. implementation of the RUN command is built into OTL; thus, dynamic pumping in

the resident executive OS is supported without revising its control program, previously derived from real-time OS by generation. This OTL expansion requires another 0.5K bytes of RAM. It should be noted that an expanded OTL can implement dynamic pumping not only from floppies, but also from magnetic tapes (provided the appropriate drivers are included in the resident executive OS at generation time). Thus, this OS supports Elektronika-60 configurations which include just magnetic tapes (cassette or conventional).

Conclusion. These principles for designing a common OS for the SM-3, SM-4 and Elektronika-60 computers were checked out in practice with positive results; this allows recommending the real-time OS as the base OS for a broad range of control systems which use this family of minis and micros. The capability of using real-time OS as the base for developing highly efficient software to be implemented on the widespread Elektronika-60 microcomputer should especially be emphasized .

The material in this article was experimentally checked by associates in the Institute of Analytic Instrument Making, NTO [Scientific and Technical Department], USSR Academy of Sciences: I. V. Bil'chuk, A. K. Vart and N. P. Korneyeva.

BIBLIOGRAPHY

1. Kibitkin, V. V., "Features of Microcomputer Operating Systems," USiM [UPRAVLYAYUSHCHIYE SISTEMY I MASHINY], No 1, 1982, pp 31-35.

2. "PDP/11 Software Handbook," Digital Equipment Corporation, Massachusetts, 1979, pp 123-165.

8545
CSO: 1863/80

UDC 681.3.06

RELBAS RELATIONAL DATA BASE MANAGEMENT SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 (manuscript received 22 Jul 83) pp 98-102

[Article by Ye. A. Barsukov, P. I. Komarov, V. A. Kurylev, G. I. Rukhamin and V. M. Ryvkin]

[Text] General description of the system. The RELBAS system is an applications program package designed for relational data base (BD) management and servicing and for organization of user interaction with it. RELBAS offers the user simple and understandable data structures, a high level of independence of applications programs of the data base schema and sufficiently simple and at the same time rather powerful data manipulation language. Some means of data manipulation and part of the data processor programs of the central executive level of the heterogeneous data base integration system were used in developing the system [1-3].

The RELBAS SUBD [data base management system] is realized in two versions, oriented toward batch processing of applications programs within the corresponding operating system: DOSYeS or OSYeS (MVT, version 6.1). Further development of RELBAS assumes support of interaction of the relational data base user in the teleprocessing mode.

The relational data base is organized on the peripheral memory in the form of an aggregation of relations [4]. The relation in the RELBAS data base management system is equivalent both in its use and in its realization of a two-dimensional table, all rows of which are identical in structure. Each row of the table is called the relation tuple and each column of the table is called the relation domain [4]. Each domain has a unique name (relation attribute) and aggregation of values of this attribute. Thus, the tuple is the sequence of values of attributes.

The data identification language (YaOD) is made available to the user for data description and organization, while the data manipulation language (YaMD) is made available for maintenance and use of data from the applications programs (PP). The data manipulation language permits one to process the relations row by row (tuple processing) and also to operate with relations as with units so as to select parts of relations (tables) that satisfy given conditions, to form new relations, to update and transform relations and so on.

Along with operators of the data manipulation language, called from the applications program, the RELBAS data base management system offers the user a number of functions which can be used autonomously. This is first of all the universal autonomous complex of data base initialization nad correction programs (UPAK), which frees the applications programmer of the need to work out his own applications programs for correction. The indicated complex can operate in initial filling modes of the data base (realization of the data base) and dynamic updating of the relational data base. Moreover, the user has the capability of using a number of autonomous service functions for printout, of the schema printing the data base relations and so on.

All the relations of the relational data base are divided into permanent relations (PtO), working (temporary) relations (RO) and so-called virtual relations (VO).

Permanent relations are permanently stored in the data base and are accessible to all applications programs operating with the data base. The user can apply any operator of the data manipulation language to the permanent relation. The permanent relation can be empty during initial work with the data base and it can be filled during operation as the data that comprises this relation is formulated; the user can correct the permanent relation. Thus, the aggregation of the permanent relation of a relational data base is the information of representation of some subject area of the real world.

If the data base administrator desires, the permanent relation can be stored in the data base in two generations, thus providing more dependable data security and integrity of data during updating. Special recovery operators of the permanent relation have been introduced in the data manipulation language for use of two generations of relations, which permit one to create the new updated generation of the same relation, thus preserving the old generation.

The working relation of the relational data base is generated by the applications program. It is placed in the working domain of the relational data base and is operational only during the operating time of the applications programs. The working relation can be, for example, the result of execution of an operation on two permanent relations, on two working relations and on a working relation and a permanent relation. The working relation is the relation which does not have to be stored and used in other programs. It is intermediate in formulation of the resulting data. The working domain for storage of the working relation can be organized both on magnetic disks (MD) and in the main memory, which permits an increase of the speed of processing comparatively small working relations.

The virtual relation is logically described in a relational data base, but is physically not stored in it. The system does not allocate a position to it in its sets of data on disk. Different sets of data, created earlier outside the system, can be included in the relational data base by using the virtual relation and new sets of data, which the user for some reason or other does not want to store in the relational data base (for example, tabulated forms printed line by line), can also be created and processed by RELBAS data manipulation languages. When working with the virtual relation, the user takes on himself the functions of organization and storage of data comprising the

virtual relation and also, if necessary, the functions of transforming this data according to the schema of the relation.

The virtual relation can be the operand of any operator of the data manipulation language. If the system is required to find the next tuple of the virtual relation, it references the user procedure (indicated in the description of the relation), which provides formulation of this tuple and returns control to the system. When the next tuple of the virtual relation is formulated (if it is the resulting relation of the data manipulation language operator), the system transfers this tuple to the indicated user procedure, which processes this tuple and transfers control back to the system.

Each relation can have a key [4], which is used in the RELBAS system, first to organize retrieval of tuples in the relation by the given values of their key and, secondly, to organize correction of the relation. The key can be complete and main. The value of the complete key clearly identifies only one tuple in a relation. The main key is formed by the first m key attributes, where $m < n$, and n is the number of attributes in a complete key. Having been given the value of the main key, one can clearly identify some aggregation of tuples in the relation.

Tuples can be stored both unordered and ordered. Tuples in an ordered (sorted) relation are written (stored) in increasing order of the attributes that comprise the schema of the given relation. Retrieval by key is possible only in an ordered relation.

The relational data base is realized as an aggregation of four direct access files on the disk. Each file is divided into logic extents, the number of which is given by the user in setting up the system. Each data base relation (permanent or working) occupies an area in one of the files, consisting of several related logic extents. The value of the logic extent is established during set up. Access to the relations in the indicated files is direct (by the name of the relation for the user and by the indication of the number of the file and the number of the first logic extent allocated to a given location inside the system). Access to the tuple within one relation is either sequential (for most data manipulation language operators) or by the value of the key (for the key retrieval operation) or by the ordinal number of the tuple in the relation (in special data manipulation language operators). The mode of index organization can be indicated in the description of the given relation to organize fast key retrieval.

Each relation should be described in the data definition language by its own schema, which determines the type and structure of the relation and its method of organization. Moreover, each attribute contained in the schema of the relation should also be described by the schema of the attribute that gives its type and format. The aggregation of relation and attribute schemas comprises the schema of the relational data base. The schema may include only descriptions of permanent and virtual relations of this data base. A subschema is given for interaction of the applications program with the relational data base, which includes:

a list of those permanent and virtual relations from the schema of the relational data base with which a given applications program will operate;

descriptions of the auxiliary working and virtual relations which are used only in a given applications program during the time of its operation and are not included in the schemas of relational data bases and descriptions of the relations are effective only within the framework of a given subschema, while new permanent relations in the subschema cannot be described;

a description of new attributes, the appearance of which is caused by the appearance of working and virtual relations in the subschema;

a description of queries to the relational data base which will subsequently be used in a given applications program when it is working with the relational data base.

The data definition language operators are selected so that they could be used to describe both the schema and the subschema. The description of the schemas and subschemas is autonomous (outside the framework of the applications program). The data definition language operators are written in special format on punched cards and have the name of the operator and a set of operands in key format. All the data manipulation language operators comprise three classes.

The first class is the algebraic operators of the data manipulation language (set theory operations on relations, projection operations, constriction operations, association operations, conditional projection operations and so on). The set of indicated operations of relational algebra has selective completeness of calculation of the relational calculus [3]. These operators permit one to use subsets of relations that satisfy the given conditions and to form new relations.

The second class is auxiliary operators (transformation and ordering of relations, computation of the function on relations and so on).

The operators of the first two classes operate with relations as whole units.

The third class is comprised of the operators of tuple processing of relations (sequential reading and writing of tuples, updating and removal of a current train, reading and writing of a tuple by a given ordinal number and also search for tuples by given values of key attributes).

The described version of the RELBAS system is a data base management system with base (host) programming language, which PL-1 or FORTRAN and also assembler can be, if agreement on connections is maintained in the language in the applications program. All the data manipulation language operators are called from the applications program by referencing the subroutine (procedure) with the corresponding name by using the CALL operator and in this case the operands of the data manipulation language operator are transmitted as parameters of the subroutine. No pretranslation of the applications program is required in this case. The subschema required by given applications program for operation of the relational data base, translated autonomously and transformed into an entity module, is connected to the applications program at the step of editing the load module.

The architecture of the RELBAS system is presented in the figure. The main components of the system are:

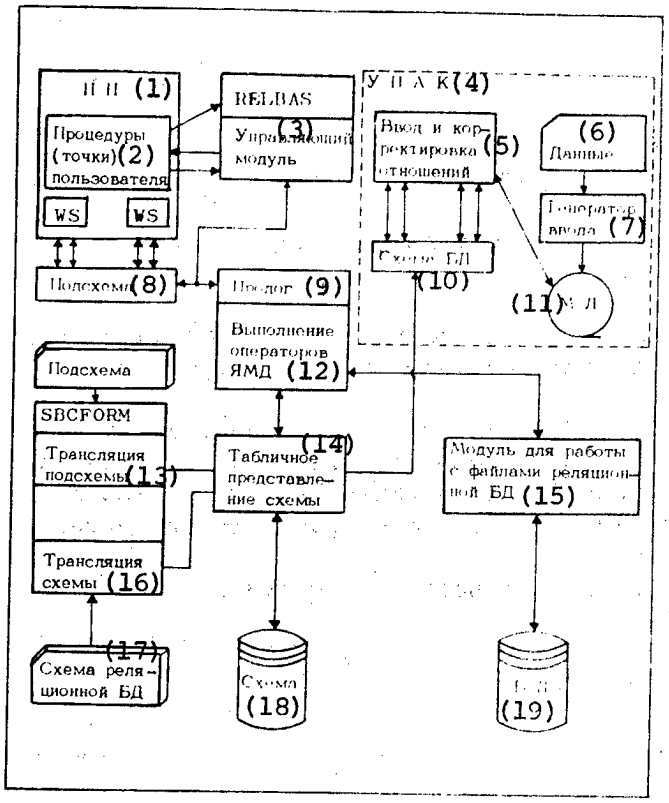
the SBCFORM schema and subschema translator;

the logic processor PROLOG for interpretation of the data manipulation language operators;

the control module of the RELBAS system;

module for working with files of the relational data base;

UPAK.



Architecture of System

KEY:

1. Applications program
2. User procedures (points)
3. Control module
4. Universal offline complex of programs for realization and correction of data base
5. Input and correction of relations
6. Data
7. Input generator
8. Subschema
9. PROLOG
10. Data base schema
11. Magnetic tape
12. Execution of data manipulation language operators

(Key continued on following page)

13. Translation of subschema
14. Tabular representation of schema
15. Module for working with files of relational data base
16. Translation of schema
17. Schema of relational data base
18. Schema
19. Data base

The relational data base itself and its schema are located in direct-access memories (on magnetic disk). A description of a schema is entered from punched cards and is translated by the SBCFORM subroutine; tabular representation of the schema is based in the schema catalog file. A description of the subschema is read from punched cards by the SBCFORM program and a tabular representation of the subschema is then formulated according to the indicated schema and the subschema description operators. The core of the RELBAS system (70 Kbyte) and tabular representation of the subschema are connected to the applications program at the editing stage. The modules of PROLOG are called dynamically in the overlay mode.

The relational data base is corrected (and realized) by using the UPAK in three steps:

formation of the subschema for correction that conforms with the schema;

entry of correction arrays using the input generator [5] and organization of correcting relations;

correction of the relations of the relational data base by UPAK devices using control cards containing data about the type of correction.

The data definition and data manipulation languages of the RELBAS system. A data definition language is used to assign the schemas and subschemas of the relational data base and only three of six operators are used in the first case. Data definition language operators are given in punchcard format.

The schema and subschema representation operator

(1) (2)

INTSC SC=<имя схемы> [, SBC=<имя подсхемы>, LANG={PL1 / FOR}]

Key:
1. Name of schema
2. Name of subschema

can be used to describe the schema and subschemas and should be the primary operator of the corresponding description. To describe the schema, the command should have only one primary operand--the name of the schema--and the two remaining operands should be absent.

The operator for representation of the relations of the schema in the subschema of the relational data base

(1)

INTRL RLS=(<список имен отношений>)

Key:
1. List of names of relations

assigns those relations from the schema of the relational data base which will be used in the given subschema.

The attribute description operator

DCLAS ASP=<имя (1) атрибута>, TYP=<тип (2)>[, LAN=<число>] (3)

Key:
1. Name of attribute
2. type
3. number

assigns the name, type and length of the value of relation attributes. The attributes can be one of four possible types:

I--integral fixed binary and its length is a word;

R--real with floating point, and its length is a word;

D--decimal (integer only);

C--character (maximum length--255).

The length of the value is given in the LAN parameter for the last two types.

The relation description operator has the following form:

DCLRL REL=<имя (1) отношения>, ASP=(<список (2) атрибутов>), TYP=(3) {P, T, V, S},
[, EXT=<размер отношения>]
[, FIL=<номер файла реляционной БД>] (4)
[, GNR=<число поколений>] (5)
[, PRC=<имя процедуры пользователя>] (6)
[, KEY=<число ключевых аспектов>] (7)
[, IND=<размер индекса>] (8)

Key:
1. Name of relation
2. List of attributes
3. Size of relation
4. Number of the file of the relational data base
5. Number of generations
6. Name of user procedure
7. Name of key aspects
8. Size of index

The TYP operand defines the type of relation: P--permanent relation of relational data base; D--temporary (working) relation on magnetic disk; V--virtual relation; S--working relation in main memory.

The size of the relation is given in the number of logic extents on the magnetic disk (for types P and T) or in bytes (for type S). The IND operand defines the existence of an index for the relation and assigns its size on the magnetic disk in extents. The compulsory PRC attribute, which indicates the name of the user procedure which is connected each time as the user point when the system reads or writes the next tuple of the virtual relation, is assigned for virtual relations.

The query description operator

(1) (2)
DCLRQ NAM=<идентификатор запроса>, DNF=<критерий выборки>

Key:
1. Query identifier
2. Selection criteria

is used only in the subschema.

A logic expression that assigns the selection criterion and is represented in disjunctive normal form (DNF) is called a query.

The query described in the subschema can then be used in different data manipulation language operators in the applications program (specifically, in the selection operator).

All the data manipulation language operators are divided into several classes:

relation initialization and opening (closing) operators;

algebraic operators on relations (including sorting and conversion operators);

relation tuple processing operators (reading, writing and updating);

relation updating operators with rewrite to a new generation and updating of the data base schema;

auxiliary operators.

The data manipulation language is formulated as a set of procedures with parameters to which one can gain access from the applications program. The operands of the data manipulation language operators are the names of the relations (R_1, R_2 and R_3), the names of logic expressions (queries) and functions, which are the logic criteria of tuple selection from the relations (DF and FUN) and the names of the working structures in the applications program (WS). The names of the relations, queries) and functions should be defined in the corresponding data definition language operators of the subschema with which the given applications program interacts.

Working structures are defined in the applications program and serve for placement of tuples selected from the relational data base. The data manipulation language operators work with the relations of all the previously considered types without restrictions. The data manipulation language operators are enumerated below:

the initialization operator of the INVOKE system (ERR), where ERR is the error and coupling domain in the applications program;

the relation open and close operators: OPENRL (<list of names of relations>) and CLOSRL (<list of names of relations>);

set theory operators for relation union, intersection and difference:

where R_1 and R_2 are the names of the operand relations and R_3 is the name of the resulting relation;

the operator of the natural joining of relations: RJOIN (R_1, R_2, R_3);

the $RESTR(R_1, \begin{Bmatrix} DF \\ R_2 \end{Bmatrix}, R_3)$ operator, depending on the type of second operand, realizes either the constrained operation of relation R_1 with respect to R_2 or the operation of selection of tuples relative to the logic condition (query) DF from relation R_1. The resulting relation R_3 can be automatically projected and converted in this case;

the relation projection operators PROJ(R_1, R_2) and conversion of the relation according to the new CONVER(R_1, R_2) schema;

the conditional relation projection operator CPROJ(R_1, R_2, R_3), equivalent to the operation of division [3] of relation R_1 by relation R_2.

The second operand can be the logic expression DF. The indicated operation is used to remove the tuples that satisfy the generality quantifier;

the function calculation operator FUNCT(R, FUN, V) confers the value of the function described in the subschema with the identifier FUN to variable V in the applications program;

the relation sort operator RSORT(R_1, R_2). In the special case, R_1 and R_2 can be the same name (so-called "sort in place");

read-write operators of next tuple of relation GETT(R, WS, AD) and PUTT(R, WS, AD), where AD is the domain of working parameters in the applications program used by the system. Use of the indicated operators in a cycle permits sequential review or filling of relations, respectively;

operators for read-write of tuple of relation according to given ordinal number N: GETN(R, WS, AD, N) and PUTN(R, WS, AS, N);

the operator for tuple by tuple update of the relation UPDTT(R, WS, AD) can be used jointly with the operator or tuple read in the cycle for sequential updating of relations. The pair of indicated operators performs so-called "updating in place";

the key search operator GETK(R, WS, AD, KEY), where KEY is the value of the key and permits search and reading of tuples according to the given values of keys. The use of the indicated operator is effective if an index is organized for the relation. Operators for deletion and updating of the tuple found by the key: DELTK and UPDTK, are related to the given operator;

the operator for updating of the relation UPDTRW(R, NPP) with rewriting of it to a new generation for correction of the relations of relational data bases with two generations, where NPP is the name of the user procedure connected during updating of each train. Moreover, the data manipulation language includes a number of auxiliary operators.

As was indicated, when solving applications tasks in the RELBAS system, one can generally do away with correction operators in the applications program, using the UPAK offline complex if it is necessary to update the data base. As shown by experience in using RELBAS in the ASU [automated control system] for power engineering, high technological effectiveness of solving applications tasks is provided in this case.

BIBLIOGRAPHY

1. Kalinichenko, L. A., V. M. Ryvkin and I. A. Chaban, "Design Principles and Architecture of SIZIF--Integrated Data Base Organization System," PROGRAMMIROVANIYE, No. 4, 1975.

2. Kalinichenko, L. A. and I. A. Chaban, "Design of Integration System of Different Types of Data Bases," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No. 4, 1981.

3. Ryvkin, V. M., "Interpretation of Relational Algebra Operations in SIZIF System," TRUDY INEUM, No. 5, Moscow, 1976.

4. Date, C., "Vvedeniye v sistemy baz dannykh" [Introduction to Data Base Systems], Moscow, Izdatel'stvo "Nauka", 1980.

5. "Generator programm vvoda dannykh dlya YeS EVM" [Data Input Program Generator for YeS Computers], Moscow, Izdatel'stvo "Statistika", 1976.

6521
CSO: 1863/122

UDC 681.3.06

GRIS INTERACTIVE GRAPHICS SYSTEM FOR MINICOMPUTER SYSTEMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 (manuscript received 15 Jun 82, after revision 22 Mar 83) pp 87-89

[Article by Gennadiy Pavlovich Demidov, engineer (Gorkiy); Vladimir Ivanovich Peskov, engineer (Gorkiy); and David Mikhaylovich Shteyman, engineer, Scientific Research Institute of Mechanics, Gorkiy State University]

[Text] Interactive design techniques are becoming more and more popular both in connection with the extension of tasks, not well formalized, handled by computers and thanks to the evolution of hardware supporting the capability of actively connecting man to the design process. Many design tasks as well as tasks of checking, processing of experiments, modeling of systems, optimization and others assume expeditious analysis of certain graphical information derived during the solution. In the process, a person must be offered the capability of rapidly changing the graphics used to display the object of investigation, and thereby actively intervene in the process of solving the problem (operation in the graphics correction mode).

Described below is the GRIS (GRaphical Interactive System), based on the SM-4 (SM-3, M-400) minicomputer and the EPG (EPG-400, EPG-SM) graphics display. It allows implementing the interactive mode of operation with graphics for a large class of applications software.

The known graphics systems oriented to similar technology have a number of substantial deficiencies and cannot fully support the requirements listed above. The GRIF [1] graphics software package does not allow using the facilities available in it for graphics display and correction together with any problem program. In the GRIf package, graphics can be corrected only after completion of the problem program, which must create a file on disk with the description of graphic objects in YaGTI [2]; then this file is converted by recoding programs into a data display format and only then can it be displayed on a screen and corrected.

The reverse process of converting data from the display format through the intermediate language into the problem program data format is similar.

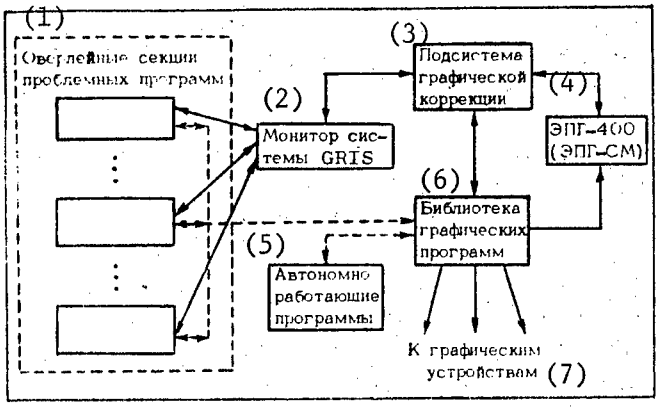


Fig. GRIS system -- user applications software interaction

Key:

1. overlay sections of problem programs
2. GRIS system monitor
3. graphics correction subsystem
4. EPG-400 (EPG-SM)
5. programs operating autonomously
6. library of graphics programs
7. to graphics devices

Thus, graphics processing is executing a chain of standalone programs, requires continually going to the operating system monitor level, is rather long in time and does not meet the requirements of efficiency.

The same deficiencies are found in the DIAGRAF interactive system, part of the base software in the ARM-R system. It allows a user to create and edit graphics on a display screen, but, just as the GRIF package, lacks facilities for direct interaction with a problem program.

In developing the GRIS system, the following principles were adhered to:
--simplicity in linking any problem program to the system;
--offering a problem program facilities as subroutines supporting display of elementary graphical objects;
--support of graphics correction mode and simplicity in branching to it from a problem program.

The scheme for interaction between the GRIS system and user applications software is shown in the drawing.

It can be seen from the diagram that a user program can function together with GRIS programs in two modes: under control of the GRIS monitor or in the autonomous mode by using library programs for graphical display. Graphics correction is possible only in the first mode. In this case, the applications software producing graphics or processing it after graphics correction is developed as an overlay section, a special type of a formatted load module.

To call these sections, there is an expandable set of directives handled by the GRIS system monitor. Correspondence between an overlay name and the directive for calling it is set by the user by using a standard editing program in a special file as a string of text

<overlay name>␣<directive name>,

where <overlay name> is the name of the file holding the load module for a given system, and <directive name> is two alphanumeric characters.

Data is transferred between the GRIS system and an overlay program through a "post box," a temporary file on magnetic disks, to which the global system parameters, current display file and other attributes are stored before the overlay call, and from which this information is restored in the program area after its call. The transfer mechanism is transparent to the user and is implemented by using two special system programs supporting the "post box" service. The only restriction on the structure of the problem program callable by the GRIS system monitor is the condition that the first executable statement in the overlay head program must be a call to the routine for reading from the "post box," and the last, a call to the routine for writing to it. The rest of the overlay section is formatted in accordance with load module requirements adopted in the operating system and is prepared independently of other overlays; therefore, linking a new problem program does not require reassembly of the entire system.

GRIS system library facilities, routines for displaying information on graphics devices, can be used in a user program In doing so, if graphics correction is not required, applications software may be formatted as a standard program module and loaded by operating system commands.

The interactive mode of operation in the GRIS system is implemented by using the graphics correction subsystem. The subsystem programs support:
--display of elementary graphical objects on the screen of the EPG-400 or EPG-SM graphics display. Elementary objects include a point, group of points, broken polygonal line, set of symbols and arc of a circle;
--control, by using the function keys, of a cursor which is used in the majority of operations for graphical correction. Discrete or continuous cursor movement is possible in one of eight directions (multiples of 45°);
--identification (capture) of a graphical object by using a light pen, confirmation or cancellation of capture;
--shift of a captured object in one of eight directions or erasure of it. A partial erasure operation is possible for a "polygonal" object ("truncating" of a line at the point where the cursor is set);
--drawing mode, i.e. construction of elementary graphical objects by using the cursor;
--combining several simple objects identifed by the light pen into one complex object for operation with it as single object;
--decomposition of a complex object, i.e. separating it into elementary components. Decomposition of an elementary broken-line object is also possible, separating the broken line into two parts at any point identified by the cursor ("breaking" of a line) or dividing it into individual segments;
--setting of system parameters (image size, cursor movement step, scale

factors and others) and parameters of graphical objects when they are constructed (line type and width, diaphragm number corresponding to the true configuration of an object displayed on the screen by a point, symbol height and slant). Default mode is possible;
--display of parameters for the system and an identified object;
--implementattion of the "window" mode, i.e. enlargement of an image in any of four quadrants with respect to any point defined by the cursor. Correction in "window" mode is a convenient facility in working with a saturated image;
--image storage on magnetic disk with capability of recall and display;
--execution of a number of service functions (erasure and display of an image fragment, element-by-element tracing of an image as a means of identifying an object when the light pen fails, gathering of "garbage"--condensing a display file-- and others.

Commands most often executed in correcting are oriented to function keys. Other commands, as a rule, which assume entry of some text, are oriented to the alphanumeric keys on the EPG-400 (EPG-SM).

The GRIS system has the capability of creating standard graphical objects and including them in a system archive. A standard object is identified by its name and can consist of one or more elementary objects positioned relative to some base point.

It is evident that the contents of the archive of standard graphical objects is governed by the set of applications tasks handled by the GRIS system. Thus, in the computer-aided manufacturing [CAM] system for printed circuit masks, standard objects displaying fitting locations of various types of microcircuits are used extensively.

The GRIS system software library is a set of program facilities for displaying elementary graphical objects on various graphics devices (the EPG-400 and EPG-SM displays; the AP-7251, AP-7252 and YeS7052 plotters; and the M-2001, M-2004 and CARTIMAT-IIIER coordinatographs). These facilities can be used directly by the problem program and are subroutines of the GRAFOR system MOVE and PLOT levels [3].

For displaying, a problem program can call subroutines which support grouping of elementary objects into a complex graphics representation which will be considered a single object when going to the graphics correction mode.

Besides the display programs, the user can employ programs for input/output of messages oriented to the EPG-400 (EPG-SM) and its alphanumeric keyboard to organize a dialogue.

Separate overlay sections operating under control of the GRIS monitor have been implemented for graphics I/O. They support:
--entry and display on the EPG-400 or EPG-SM screen of graphics prepared on the EM-709, UKPP-3 or UKPP-4 coding devices, or information in the codes of the M-2001, M-2004 and CARTIMAT-IIIER coordinatographs;

--entry and display on the EPG-400 or EPG-SM screen of graphics written in YaGTI;
--graphics output from the EPG-400 or EPG-SM screens to the AP-7251, AP-7252 and YeS7052 plotters;
--graphics output from the EPG-400 or EPG-SM screens to perforated tape for the M-2001, M-2004 and CARTIMAT-IIER coordinatographs;
--graphics translation into the YaGTI.

GRIS system software has been implemented in FORTRAN with minimal use of Macro Assembler and functions under control of the RAFOS operating system.

The total size of the GRIS system software, including the graphics correction subsystem, libraries of display routines and overlay sections for graphics I/O, is now about 600K bytes; but the system can function fully with SM-4 (SM-3, M-400) computers with 48K bytes of main memory; this is achieved through efficient use of program overlaps within each overlay section.

The interactive GRIS system has been in operation for two years on the ARM-R hardware system. Rather high reliability, easy configurability, and large set of editing facilities have allowed using it as a base in developing a number of interactive systems and software packages, such as a package for making photomasks for plates with inscriptions (nameplates), a system for editing photomasks for printed circuits, and a system for making photo originals of micro assemblies.

GRIS system facilities are also used in a package of software for modeling and analysis of REA [radioelectronic device] assemblies and in a system for automated layout of printed circuits.

BIBLIOGRAPHY

1. Yelshin, Yu. M., "Using the ARM-R [Automated Workstation] for Printed Circuit Design," OBMEN OPYTOM V RADIOPROMYSHLENNOSTI, No 10, 1979, pp 1-9.

2. Vermishev, Yu. Kh.; Yelshin, Yu. M.; Bibikov, V. F.; and Lomakina, Z. I., "On the Structure of Graphics and Text in Computer-Aided Design Systems," OBMEN OPYTOM V RADIOPROMYSHLENNOSTI, No 2, 1977, pp 22-23.

3. Bayakovskiy, Yu. M.; Mikhaylova, T. N.; and Mishakova, S. T., "GRAFOR: kompleks graficheskikh programm na FORTRANe" [GRAFOR Graphics Software Package in FORTRAN], Moscow, 1972, 60 pages (IPM AN SSSR [USSR Academy of Sciences Institute of Applied Mathematics], Preprint No 5).

8545
CSO: 1863/80

COMPUTER ASSISTED AGRICULTURAL MANAGEMENT

Moscow EKONOMICHESKAYA GAZETA in Russian No 4, Jan 84 p 18

[Article by P. Arkhipov, director of the Latvian computer center and candidate in economic sciences: "The Computer Is Coming To Our Assistance"]

[Text] A major effort is underway in our republic to develop and introduce OASU-sel'khoz [branch automated management system for agriculture]. This effort is based on a unified coordination plan and has enlisted the assistance of the LRVTs KP SKh [Latvian Republic Collective Use Computer Center for Agriculture] (as chief designer), all branch institutes of the republic's Ministry of Agriculture, the Latvian Academy of Agriculture, the Institute of Economics of the LaSSR Academy of Sciences and other republican scientific and design organizations.

The first step of the OASU project focused on establishing a system of day-to-day administration in the khlkhozes, sovkhozes and RAPO. The dispatching service has become the technical and organizational basis for this, it is operating at each kolkhoz and sovkhoz, at each RAPO and in the Ministry of Agriculture. All units of agriculture--from the brigade and farm up the Ministry--are tied into the operational connection.

During the past few years, the dispatcher service has made wide use of micro- and minicomputers. In the Talsinsk RAPO, for instance, all farms are equipped with ISKRA-534 computers. The SM-4 computer is used to process the summer operational data and to do the analysis.

The introduction of the dispatching control has allowed the management of agricultural jobs to be carried on efficiently and in a centralized manner. It also permits the wide application of the Ipatovsk method of machinery utilization. Eight-hundred full detachments worked in the 1982 harvest of course fodder in the republic. They fulfilled the plan by 104 percent.

Along with the introduction of the system of operational production management, there is also a major project underway to establish an automated system of plan calculations. Such a system allows the calculation of up to 10 versions of a plan. Each version has different criteria of optimality. They can be calculated by the maximum of net or gross profit, by the minimum

of production costs and by other indicators. For this, it is necessary to have a large quantity of scientifically sound, standardized reference information. Suffice it to say that in order to put together a long-range plan of agricultural development for a kolkhoz or sovkhoz, from 4 to 5 thousand pieces of informations are needed.

Together with economic specialists and RAPO, specialists at the Latvian computer center are preparing normative data for drawing up and compiling long-range plans. This involves primarily such important normative indicators as crop capacity, livestock productivity, labor costs, etc. Of course, such consolidated normatives--so-called variables--require a scientifically valid approach to their design, taking into consideration soil type, climatic zone and other objective production factors. The mathematical methods of analysis used by us and scientists from the branch institutes made it possible to determine which factors contribute to an increase in crop capacity and livestock productivity and to establish their value in economic terms.

9992
CSO: 1863/121

UDC 681.3.01

DISKRET COMPUTER NETWORK BASED ON PACKET RADIO LINK

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
(manuscript received 10 Mar 83) pp 37-44

[Article by Andrey Mikhaylovich Luchuk, doctor of engineering sciences, UkSSR Academy of Sciences Institute of Cybernetics; Rafail Grigor'yevich Ofengenden, doctor of engineering sciences, UkSSR Academy of Sciences IYaI [Institute of Nuclear Research]; Sergey Georgiyevich Bunin, candidate of engineering sciences, UkSSR Academy of Sciences Institute of Cybernetics; and Anatoliy Petrovich Voyter, engineer, UkSSR Academy of Sciences IYaI [Institute of Nuclear Research]; all from Kiev]

[Text] Introduction. The current status of computer networks is described by an increasing level of requirements for the technical and economic parameters of communications channels. This has led to the search for new, non-traditional ways of solving the problem of designing a data communication network.

Thus, at the start of the seventies, the capability of using a ground, packet radio link for remote user access to central computer complex resources was demonstrated [1]; this marked the beginning of numerous studies in this field.

This article concerns the regional (within a city) computer network Diskret [2], a fragment of which is now in trial operation. The Diskret network is being developed in phases. In the first phase, the basic principles of network functioning are being drafted as applied to specific conditions, statistics are being compiled, parameters are being refined to implement a standard hardware and software system supporting connection of different types of subscribers to the network.

Prerequisites and Problems of Network Development. Arguments in favor of developing a computer network using a packet radio broadcast link are based on purely practical considerations. The technical characteristics of existing public wire networks do not meet the goals of developing a network of computer centers [3]. But building special, broadband, data communication networks based on cable, waveguide, radio-relay or optical fiber communication circuits requires an investment of considerable resources and time. Therefore, under established conditions, it is advisable to use a ground packet radio link at least for building regional computer networks, for example, on the scale of a major city, where a significant share of information streams originates and is used and the need for distributed processing of information is great.

By analyzing and comparing a ground packet radio link with other types of communication, one can identify a number of advantages inherent to it, among which let us note primarily the:
--capability of developing a fully connected data communication network with high throughput and reliability over considerable territory;
--relatively low costs for development and operation;
--simplicity of expanding the network both in number of subscribers and in territory covered;
--speed and convenience in setting up; and the
--capability of including mobile subscribers and those who change their location periodically in the network.

We must also note some shortcomings of a packet radio link, such as complexity in practical implementation under conditions of spectrum overloading, the potential possibility of unauthorized access to information circulating in the network, and the possibility of creating intentional interference. These problems are not fundamental. Use of appropriate types of modulation [4] and various types of cryptographic systems [5] allows eliminating them to a considerable extent, if not completely.

There are several foreign packet-switching radio networks now in operation [1, 4]. In developing the domestic Diskret computer network, two main aims are being pursued:
--further scientific and technical research in computer network architecture, software and hardware with a ground, packet radio broadcast link;
--combining the information and computing resources of various institutions within the UkSSR Academy of Sciences on a city-wide scale for joint use of these resources primarily for automation of scientific research.

To reach these goals, the following basic problems must be solved:
--selection of a frequency band, structure of signal and energy potential of the radio channel;
--selection of the data communication structure;
--research and development of inexpensive and reliable technical components for the physical channel;
--development of a standard network adapter for connecting various subscribers to the network;
--development of software for efficient functioning of this type of network and convenient forms of user interaction with the Diskret and other networks.

Discussed below are the principles for implementation and specific facilities developed within the scope of solving these problems, except for the last one which will be discussed separately.

Network Logic Structure. The Diskret computer network is based on the open system architecture suggested by the International Standards Organization (ISO) [6]. In accordance with this, the Diskret network is a set of logic elements (systems) joining a set of special hardware and software into an open information and computing environment in an area limited to the operating radius of the ground UKV [ultrashort wave] radio link.

By functions performed, the network will contain systems with these specializations: operating (offering the resources), terminal (using these resources), administrative (acquiring statistics and issuing network information), and converter (interacting with other networks). There is also a communications system containing the unique center for packet switching in the form of a radio channel with the communication modules for each system connected to it.

Each system, except the communications, must have a hierarchical structure [7] differing somewhat from that adopted by the ISO. The differences are caused by communications system features: first, absence of the network level, and second, splitting of the second level into two sublevels: information channel control and control of multiaccess to the radio channel.

The network level is excluded because there is no need of handling the corresponding functional task, namely: routing packets when they pass through the communications network. But considering the joint use of a common data communication channel by the systems, operating moreover without mutual coordination, the problem of determining the time when packets can be sent has to be solved in each of them. The access control sublevel affords service for the channel sublevel and in turn uses the services of the physical level. This functional task is allocated to a separate sublevel in the system structure because of the considerable range in complexity of this task, from the simplest procedure of carrier-sense multiple access [CSMA] [8] to various types of adaptive procedures with dynamic priorities. Use of a particular access method is determined by the level of network development and nature of processes performed in it.

Three virtual networks embedded in each other, the communications, transport and computing, form the logic structure of the Diskret network. The communication network must ensure reliable transmission of data packets between various systems by a collectively-used band of radio frequencies under the conditions of the presence of noise on the channel and possibility of packet collision when transmissions overlap in time. The logic elements in the communication network are the radio channel and communication modules formed by the set of the two lower levels in the structure in each system. The transport network, by using the communication network facilities, enables transmission of data, encoded by any method, between processes by setting up logic channels between their ports. In doing so, the process can be an applications program as a whole, a point in this program using or generating data, terminal operator and others. The transport network is implemented by the set of transport modules in each system in which the functions of the fourth level in the ISO structure are performed. The computer network enables interaction between processes in various systems through standard logic channels, forming a virtual network of processes in doing so. The processes in each system are represented by the three upper levels of its structure.

Network Physical Structure. The physical structure of the operational fragment of the Diskret network is shown in fig. 1. It follows from it that the two operating frequency bands, retransmitter, subscriber transceivers, antennas and feeders, and modems form the physical structure of the first element in the communication network, the radio channel.

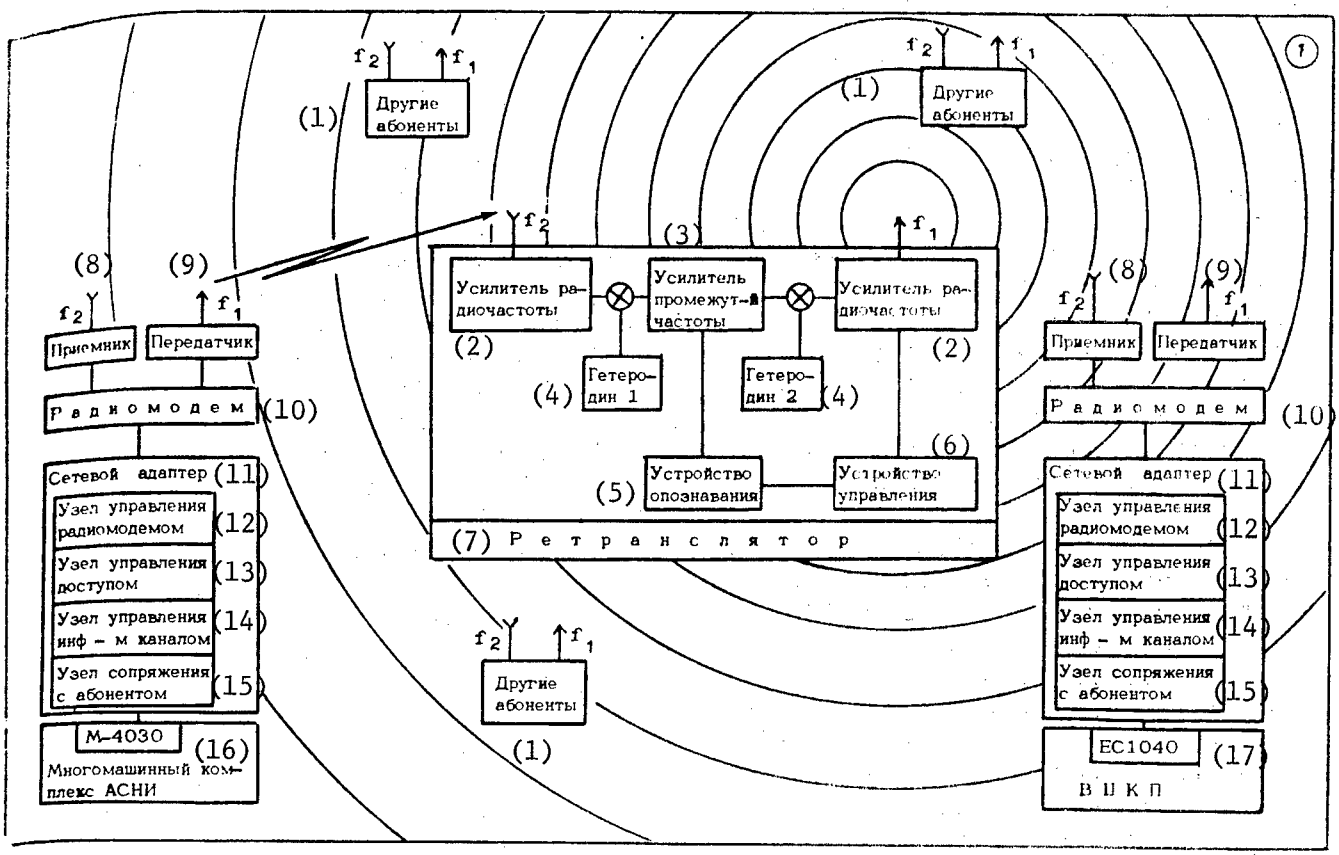


Fig. 1. Diskret network physical structure.

Key:

1. other subscribers
2. radio frequency amplifier
3. intermediate frequency amplifier
4. heterodyne 1, 2
5. recognition unit
6. control unit
7. retransmitter
8. receiver
9. transmitter
10. radio modem
11. network adapter
12. radio modem control unit
13. access control unit
14. information channel control
15. subscriber interface
16. M-4030, multimachine complex in automated system for scientific research
17. YeS1040, multiuser computer center

In developing the network, parameters such as data transmission rate, carrier frequency, band of frequencies used, keying type and energy potential level were selected on the basis of their optimization, proceeding from existing capabilities for frequency range allocation, maximal simplicity of transceivers and minimal power of subscriber transmitters. As a result, the capability of covering an area of about 8,000 square kilometers by the Diskret

network was obtained and with that, implementing data transmission at a rate up to 100K bits/s with a validity no worse than 10^{-6} in the 150 kHz frequency band. The decision to use frequency-shift keying [FSK], although inferior in the signal-to-noise ratio [SNR] to phase-shift keying [PSK] by about 2 dB, allows considerable simplification of the signal processing centers. Operating frequencies in the decimetric band were selected. This band features limited range. Here we have in mind the mechanism for radio wave propagation by direct or almost direct line of sight, since other propagation mechanisms (ionospheric, tropospheric and others) do not afford the required throughput and reliability. The geographic layout of potential network subscribers and terrain relief shaped the need for using a retransmitter set up within limits of radio visibility of all subscribers. In connection with this, the problem of selecting one of two possible alternatives for the data communication network structure arises: with one band of operating frequencies or with separate frequencies for sending and receiving.

In the first alternative, the retransmitter, in its composition, must have a one-packet buffer to separate the process of receiving and sending in time. It also must perform certain functions related to monitoring packets being received so that damaged packets are not retransmitted in order to increase radio channel throughput. In the second alternative, the retransmitter structure is considerably simplified since this way sending and receiving can overlap in time; thus, the packet buffering process can be eliminated. It is obvious that the time for sending packets of equal length is identical for both structures under consideration. From here on, let us assume that data is communicated at the maximum rate permitted in the frequency band used.

In the second alternative, the so-called hidden terminals problem is automatically solved. But in the first alternative, special measures have to be taken to solve it since when the retransmitter receives the next packet, subscribers hidden from the preceding one will assume from the absence of a carrier signal that the radio channel is free and that they may begin sending when necessary; this will cause overlap in time and collision of packets. Two ways to solve the hidden terminals problem in this network structure are:
--allocating from the available band of operating frequencies a narrow-band subchannel for sending a "busy" tone by the retransmitter while a packet is being received. This method is analyzed in detail in [9]. Depending on the width of the "busy" subchannel, it affords a normalized effective transmission rate up to 0.72;
--introducing a special procedure signaling all subscribers of the start of reception by the retransmitter of the next packet.

The latter can be done, for example, like this: The retransmitter turns its transmitter on when a packet appears at its input (determined by the carrier signal) for the brief time τ, normalized according to the time for sending one packet. All subscribers in receive mode record the appearance of this carrier pulse and will decide that the retransmitter is busy for the immediate time equal to $2a + \tau + 0.5$ and will not try to start sending. Shown in fig. 2 is a time diagram of the radio channel operation by this procedure. The symbols used here will be explained when the quality of both structure alternatives in

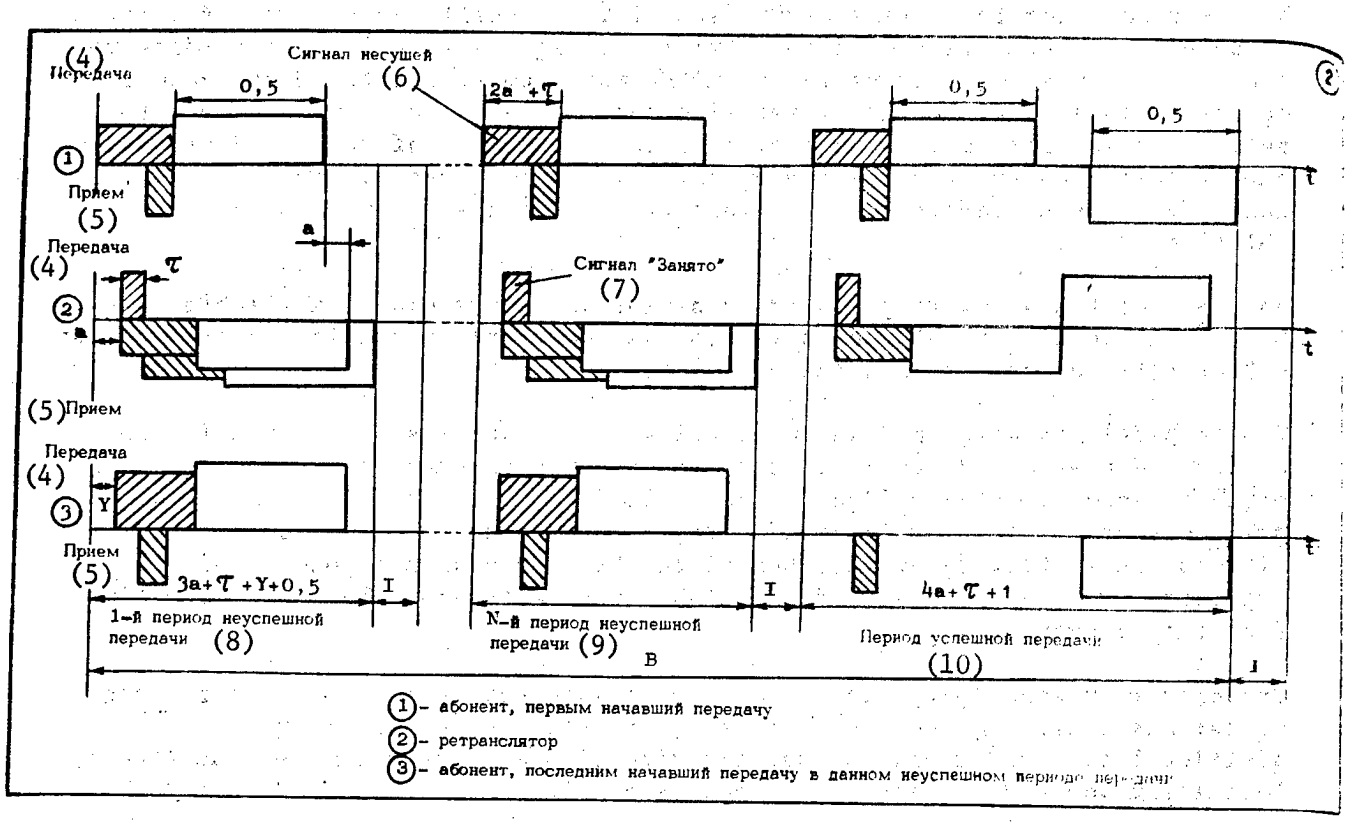


Fig. 2. Time diagram of radio channel operation using special procedure.

Key:

1. subscriber who first started sending
2. retranmsitter
3. subscriber who last started sending in a given unsuccessful period of transmission
4. transmission
5. reception
6. carrier signal
7. "busy" signal
8. first period of unsuccessful transmission
9. N-th period of unsuccessful transmission
10. period of successful transmission

question is analyzed. To keep a packet sent by a subscriber secure when the retransmitter transmitter is turned on for time γ, the start of a packet must be preceded by a carrier signal without keying with length $2a + \gamma$. A particular alternative for the data communication network structure applicable to specific conditions is chosen on the basis of comparative assessment of their efficiency which is discussed below. But here let us note that separate send and receive frequencies have been adopted in the Diskret network. All subscribers, using a pencil-beam antenna, are linked only to the retransmitter. Subscribers send on frequency f1. All subscriber receivers are tuned to frequency f2 for operation of the retransmitter transmitter, which is linked to all network subscribers by an essentially circular-beam antenna.

The retransmitter is a transceiver (see fig. 1), shifting the band of receive frequencies by some value and amplifying the signals to a level affording reception with the required validity in the most remote points in the network operating zone. The retransmitter also has an identification unit (UO) for network subscriber signals to prevent unauthorized use of the retransmitter or amplification and retransmission of any noise on its receive band. A control unit (UU) enables remote switch-on/off of the retransmitter, and thus the entire network.

Analysis of transceivers produced in series and commercially available combined with network requirements led to the necessity of special development of subscriber transceivers and radio modems.

The transceivers were made as small units and can be installed both in the network adapter and directly at the subscriber's antenna to minimize loss of high-frequency energy in feeder lines. In doing so, special attention was paid to ease in manufacturing and cost. In the operating version, the transmitters are a discrete synthesizer of the frequency with a power amplifier. The receivers are built according to the superheteodyne scheme with one-time frequency conversion.

Subscriber antenna design is a dipole with a corner reflector. The retransmitter has a circular antenna with the lobe sent to the ground which yields additional amplification. A specific directional pattern can be generated by installing additional passive reflectors or directors.

A radio modem is used to convert an arriving binary series into a frequency-modulated [FM] signal when sending and to restore this series when receiving. For this, it has a frequency modulator, frequency detector, bit synchronizer based on automatic phase-lock loop, and a device for compensation of biases, caused by transmitter frequency instability and shift of the direct component when keying codes are changed. There is also a carrier detector and network adapter interface which executes procedures specified in the CCITT X.21 bis recommendation [10].

The network adapter (as a communication network element) implements the lower two levels of the structure in each system. In accordance with this, its basic functional units are the radio modem control unit, the access control unit and the information channel control unit. The adapter also includes a subscriber interface unit (USA) which implements mechanical, electrical and procedural matching with the subscriber interface connected to the network. The functions of this unit are not reflected in the logic structure since in essence they belong to the interlevel interface.

The algorithm for operation of the modem control unit is based on the CCITT X.21 bis recommendation [10]. In accordance with this recommendation, data is transmitted in a serial synchronous manner. A feature is the use of the "Received Line Signal Detector" circuit to indicate a carrier on the radio channel which is required for operation of the access control unit.

Of major importance to normal network operation is the access method adopted and common to all subscribers. The access control unit handles the task of ordering transmissions in a common band of operating frequencies to completely preclude or reduce to the minimum the number of transmissions overlapping in time (or duration of their overlap). The criterion of quality of the access method from the view of efficient use of the spectrum is the effective communication rate, while from the subscriber's view, it is the delay in packet transmission. There is a set of methods (protocols) for dividing the common communication channel among subscribers. They can be classified as follows:
--fixed function protocols (frequency or time division);
--dynamic backup protocols (both with centralized and decentralized control);
--random access protocols.

A pulsing schedule is typical of computer network suscribers, i.e. the peak value substantially exceeds the mean level. The degree of use of a communication channel in a pulsing schedule increases as one moves down the list of protocol types. There are essentially two classes of random acccess protocols [11]: ALOHA type protocols based on complete lack of situation monitoring in a network, and the carrier-sense multiple access [CSMA] protocols.

The latter are used in the ground radio link, where signal propagation delay in the network is relatively slight. The data communication channel structure adopted in the network allowed using an access protocol based on carrier sensing before sending and auditing the signals proper while sending [12]. It is unacceptable in the structure version with a common frequency band since the transmitter signal overloads the receiver proper, which precludes the capability of determining the fact of collision of packets in its sending process. Thus, selection of the multiaccess protocol to a certain extent is due to the radio channel structure. The access control unit implements the protocol procedures in the "non-persistent" mode [11], i.e. when a subscriber has a packet ready to send and detects a busy radio channel, he does not wait "persistently" until the channel is free, but puts off the retry to send until a later time allocated randomly. This reduces the probability packet uverlap and thus improves channel usage.

Of practical interest is defining the quality of this protocol under the conditions of the Diskret network in terms of "effective rate-delay." For this, let us use a model with an infinite number of users combined with renewal theory methods [11]. As follows from the time diagram for the radio channel operation (fig. 3), its status can be represented as a series of nonintersecting cycles of successful transmission of one packet with average value B and idle time with mean I. In turn, a successful transmission cycle consists of a random number of unsuccessful transmission periods with mean value N, separated by idle periods. Thus,

$$B = N\,(4a + c + Y + I) + 1 + 2a,$$

where a and c are, respectively, the maximal time for propagating a signal from the subscriber to the retransmitter and the time of indication of collision of packets, normalized by T (packet transmission duration); Y is the mean time of start of sending the last packet overlapped by the one being sent. After setting T = 1, let us define the effective communication rate as the

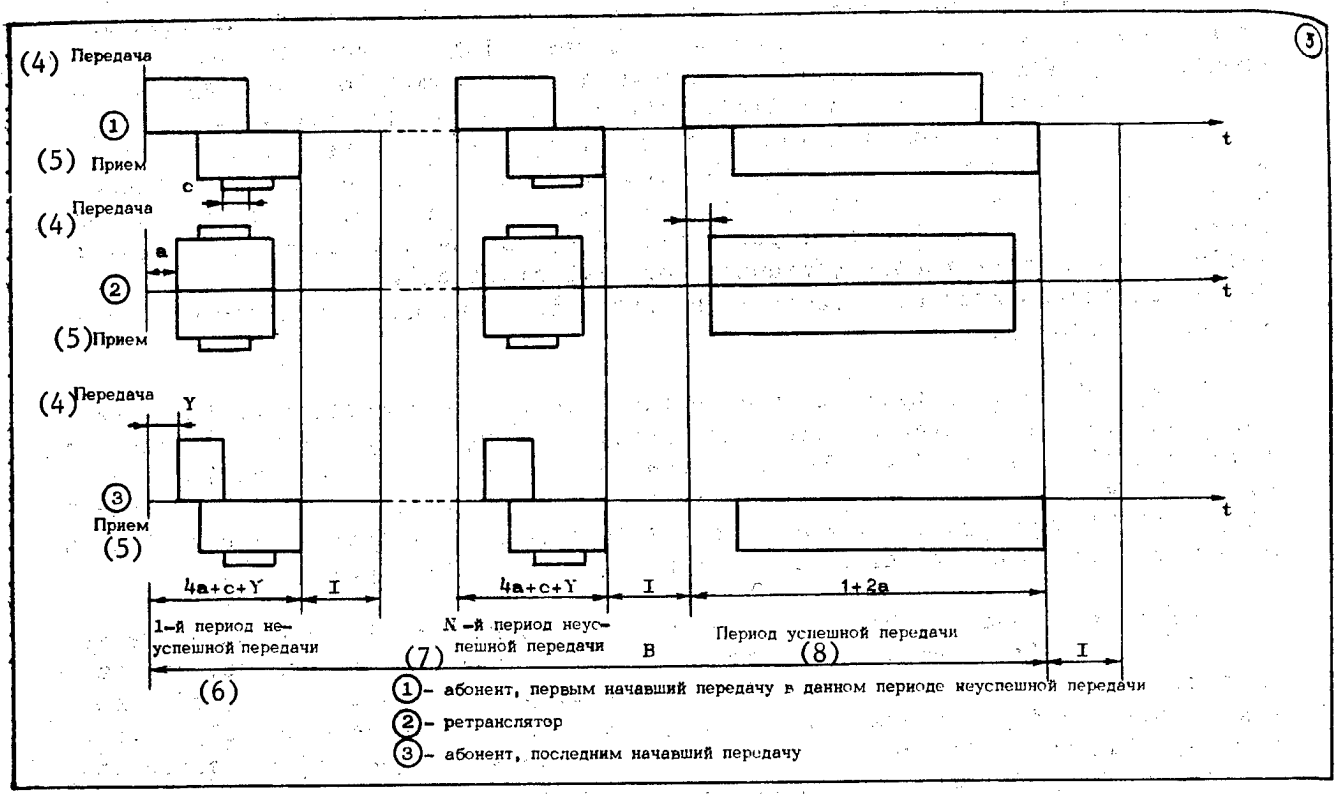


Fig. 3. Time diagram of radio channel operation.

Key:

1. subscriber who first started sending in a given period of of unsuccessful transmission
2. retranmsitter
3. subscriber who last started sending
4. transmission
5. reception
6. first period of unsuccessful transmission
7. N-th period of unsuccessful transmission
8. period of successful transmission

share of time from the sum of lengths of successful transmission cycles following one after the other and idle time, when a packet is sent without conflict:

$$S_2 = \frac{1}{B + I}$$

From here on, the subscript 1 or 2 indicates the version of the data communication network structure with one band of operating frequencies or with separate send and receive frequencies, respectively. The effective rate actually reflects the rate of communication through the radio channel in packets in

time T. It is evident the maximum value of S = 1 is possible only in the ideal organization of multiple access and the corresponding traffic level, when the packets are sent without overlaps and gaps between them. Considering that the number of periods of unsuccessful transmission within a successful transmission cycle is distributed geometrically with the mean

$$N = e^{2aG} - 1,$$

and considering these values defined in [11]

$$I = \frac{1}{G} \text{ and } Y = 2a - \frac{1 - e^{2aG}}{G},$$

let us finally derive:

$$S_2 = \frac{G}{(e^{2aG} - 1)[(6a + c)G + e^{-2aG}] + G(1 + 2a) + 1},$$

where G is the summary (of both new and repeat packets) network traffic rate, normalized by T. The expression derived yields a lower estimate of the effective transmission rate since it assumes identical distance of all subscribers from the retransmitter equal to maximal.

Let us define the mean delay of packet transmission for each subscriber as the mean time from packet origination to completion of its successful transmission. With regard to assumptions made in the model, and the number of unsuccessful send attempts being distributed geometrically with the mean $(\frac{G}{S_2} - 1)$, we can derive an expression to estimate the mean sending delay:

$$D_2 = \left(\frac{G}{S_2} - 1\right)(5a + c + Y + X) + 1, \qquad (1)$$

where X is the mean time of delay of a repeated transmission.

In a similar way (with regard to fig. 2), let us define the corresponding characteristics of the adopted protocol for the structure with a common band of frequencies:

$$S_1 = \frac{G}{(e^{2aG} - 1)[(5a + \tau + 0{,}5)G + e^{-2aG}] + (4a + \tau + 1)G + 1},$$

$$D_1 = \left(\frac{G}{S_1} - 1\right)(4a + \tau + 0{,}5 + Y + X) + 4a + \tau + 1 \qquad (2)$$

Fig. 4 shows the quantitative assessment of the effective transmission rate for the alternatives discussed for the data communication network structure with different values of traffic and real a, c and τ. A comparison of (1) and (2) shows the advantage of choosing the structure with separate send and receive frequencies (fig. 5).

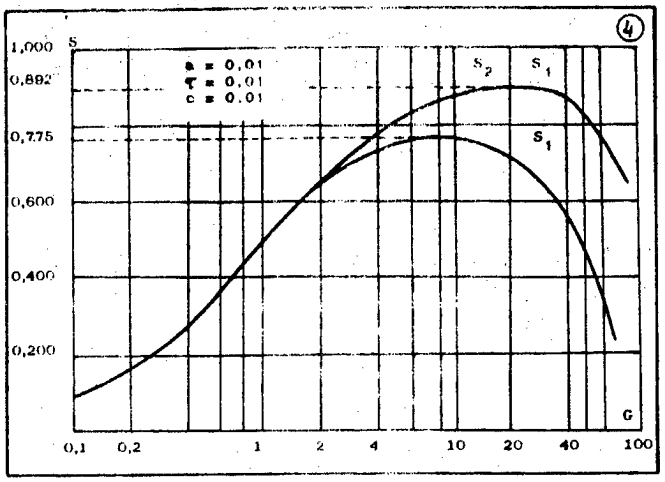


Fig. 4. Quantitative assessment of effective transmission rate for the alternatives discussed for the data communication network structure with different values of traffic and real a, c and τ.

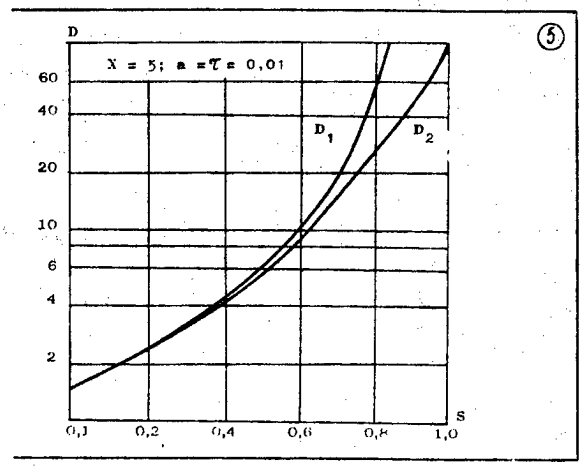


Fig. 5. Comparison of (1) and (2) showing advantage of choosing structure with separate send and receive frequencies.

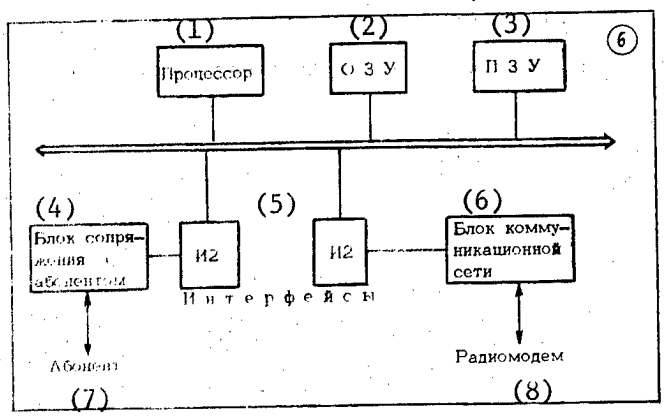


Fig. 6. Network adapter based on Elektronika-60 microcomputer.

Key:

1. processor
2. RAM
3. ROM
4. subscriber interface unit
5. interfaces, I2
6. communication network unit
7. subscriber
8. radio modem

The main task of the information channel control module is to ensure reliable data transmission at the packet level through the communication network. There are problems in selecting the type of protocol implemented by this module. The fact is: Subscriber interaction in the network on an equal-rights basis is possible only when there are two address fields in the format of the packets to be sent: one for the originator and one for the destination. Packet formats with two address fields are typical in protocols for local networks developed in various forms. Unfortunately, today there is no generally accepted concept for this protocol. In the Diskret network, in the initial phase of development, it was decided to use the Unified System standards, procedures and data teleprocessing software, particularly the BSC [Binary Synchronous] byte-oriented procedure for information channel control, in which the packet format has these address fields. Reception, selection, generation and issuing of these fields are performed by the network adapter autonomously, which avoids the need of making changes to the standard software [13, 14].

Components in the operating fragment of the Diskret network are at the breadboard level. Thus, in the network adapter, the functions of subscriber interface, radio modem control, access control and certain information channel control functions have been implemented by circuits. The subscriber interface is oriented only to the Unified System I/O interface.

Within the scope of further network development, a network adapter has been designed which is based on the Elektronika-60 microcomputer; its structural diagram is shown in fig. 6. Using the microcomputer will allow implementing network adapter structures that take into account the functional capabilities of subscribers connected to the network (YeS EVM, SM EVM, BESM-6 and others), and developing network subscriber stations on its basis. In terms of the model of the network logic structure described, all system levels must be implemented in the network adapter in the latter case. In the network operating systems, two alternatives are suggested for adapter structure in which two or three lower system levels, including the transport, are fully covered. This will allow unburdening the computer of communication network control functions to a considerable extent.

The subscriber interface unit (BSA) and corresponding program module in ROM are replaceable and are determined by the type of I/O interface for a particular subscriber. The communication network unit (BKS) together with its program module is identical in all network adapters and performs the procedures for radio modem, multiple access and information channel control. Both units connect to a common bus through the standard I2 parallel exchange unit.

Conclusion. The ways considered for solving problems related to the first phase in setting up the regional computer network based on a packet radio broadcast link, and the hardware developed in the process, have laid the foundation for the operating fragment of the Diskret network. Experience in operating this fragment has shown ways of imrpoving the engineering solutions and has allowed formulating the basic requirements for software implementing (with regard to network architectural specifics) the procedures for user interaction with the network.

BIBLIOGRAPHY

1. Abramson, N., "The ALOHA System: Another Alternative for Computer Communications," PROC. AFIPS: FALL JOINT COMP. CONF., Vol 37, 1970, pp 281-285.

2. Luchuk, A. M.; Ofengenden, R. G.; Bunin, S. G.; and Voyter, A. P., "Experimental Computer Network Based on a Packet Radio Braodcast Link," in "Vychislitel'nyye seti kommutatsii paketov: Tez. dokl. 2-y Vsesoyuzn. konf." [Packet Switching Computer Networks: Theses of Papers from Second All-Union Conference], Riga, Vol 2, 1981, pp 94-98.

3. Marchuk, G. I. and Kotov, V. Ye., "Problems of Computer Engineering and Basic Research," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, No 2, 1979, pp 3-14.

4. (Kan, R. E.; Grounmeyer, S. A.; Berchfil, J.; and Kanzelman, R. S., "Progress in Packet Radio Communications," TIIER, Vol 66, No 11, 1978, pp 212-247.

5. (Hoffman, L. J.), "Modern Methods of Protecting Information," Moscow, Sov. radio, 1980, 263 pages.

6. "ISO/TC 97/SC 16. Reference Model of Open Systems Architecture (Vers. 3), 1978, pp 1-84.

7. Yakubaytis, E. A., "Arkhitektura vychislitel'nykh setey" [Computer Network Architecture], Moscow, Statistika, 1980, 279 pages.

8. Kleinrock, L., "Computer Systems with Queues," Moscow, Mir, 1979, 600 pages.

9. Tobagi, F. and Kleinrock, L., "Packet Switching in Radio Channels. Part 2. The Hidden Terminals Problem in Carrier Sense Multiple Access and Busy Tone Solution," IEEE TRANS. ON COMMUN., 1975, pp 1417-1433.

10. "Orange Book, Vol 8, 2. Public Data Network," CCITT Sixth Pleanry Assembly, Geneva, 1977, 218 pages.

11. Kleinrock, L. and Tobagi, F., "Packet Switching in Radio Channels. Part 1. Carrier Sense Multiple Access Modes and Their Throughput-Delay Characteristics," IEEE TRANS. ON COMMUN., 1975, pp 1400-1416.

12. Hopkins, G., "Multimode Communication on the MITRENET," COMP. NETWORKS, No 4, 1980, pp 229-233.

13. Ofengenden, S. R. and Nazarova, Ye. I., "Data Teleprocessing Software for Radio Channels," in "1-y Vsesoyuzn. semin. po avtomatizatsii nauch. issledovaniy v yadernoy fizike i smezhnykh oblastyakh: Tez. dokl." [First All-Union Seminar on Automating Scientific Research in Nuclear Physics and Related Fields: Theses of Papers], Dushanbe, 1980, p 203.

14. Serykov, G. S., "Teleprocessing Software for Subscriber Computer Network," USiM, No 1, 1981, pp 74-77.

8545
CSO: 1863/80

ORGANIZATIONAL AIDS, HARDWARE AND SOFTWARE FOR MANAGEMENT OF COMPUTING PROCESS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 pp 122-124

[Article by B. N. Pan'shin]

[Text] The seminar "Organizational Aids, Hardware and Software for Management of the Computing Process," organized by the Republic House of Economic and Scientific and Technical Propaganda, the Institute of Cybernetics imeni V. M. Glushkov, Ukrainian SSR Academy of Sciences, and by the Exhibition of Achievements in the National Economy of the Ukrainian SSR, was held in Kiev on 20-24 September 1983.

According to established tradition (the seminar has been conducted annually in September since 1978), problems of creation, development and operation of organizational aids and software for management of data processing techniques at VTsKP [Collective Use Computer Center] and computer networks were considered at the seminar. It was noted that a modern computer center is a complex technological system, which is developed under the influence of ever more rigid requirements on productivity, reliability and efficiency of using hardware and software.

The tendency toward improvement of the technology of offering different services to the mass user (due to introduction of time-sharing and teleprocessing modes), manifested in rapid restructuring of the technological data processing system, a change of the relative significance of its individual elements and in replacement and modification of them, is especially typical and significant for modern VTsKP and computer networks. All this in turn determines the development of the hardware of computer centers, improvement of the software and information and organizational support of systems for management of the computing process, and the nature of these changes is so dynamic and the number of developments of programs for management of the computing process is growing so rapidly that an annual seminar of this type, in the opinion of its participants, is quite justified.

As in previous years, the following four directions dominated the work of the seminar:

development and operation of software for automation of the control functions of organizational and operational processes at a modern computer center (VTsKP);

development and operation of software for management of the techniques of solving applied problems;

improvement of existing and development of new organizational aids and hardware for the computing process;

software tools.

The need for coordinated consideration of these problems was emphasized in all the reports and communications (a total of more than 40) with regard to the interrelationship and mutual influence of hardware, software and organizational support of computer centers on each other. The considerably increased influence of organizational support and software for automation of the organizational functions of management of the computing process ("payment" due to the growth of the productivity of computer systems) was noted. The opinion that the development of software for management of the computing process (in combination with development of organizational support and organizational hardware) is an independent direction in the overall problem of development and improvement of the technique of automated data processing based on VTsKP and computer networks, was refined and again confirmed during discussion of the reports.

A number of reports weredevoted to generalization of the experience of development and operation of software for automation of control functions of organizational and operating processes at a modern computer center (VTsKP), among which the following evoked the greatest interest: "Organization of Mutual Calculations With VTsKP Users When Using Remote Terminals in the Time-Sharing Mode" (A. R. Myagi, Tallin), "A Resource Planning and Accounting System in YeS Computer Systems" (S. V. Makhnayev, L. B. Gradus and A. L. Alayev, Moscow), "The Distributed Administration System of VTsKP" (A. F. Tutov, S. G. Kusrashvili and V. S. Trokhimenko, Kiev), "Software for Maintenance of Magnetic Carrier Archives" (S. G. Serykov, Kiev) and "A Program Complex for Classification and Storage of Data on Breakdowns of Devices of YeS Computers" (L. K. Snigireva, Kiev).

The evaluations, opinions and questions on these reports, advanced by the seminar participants, showed that the following directions were most clearly noted in the overall problem of software development of automation of computer center personnel work:

development of software for organization of effective interaction in the "computer-user-VTsKP administration" chain (for automation of the work of the computer center administration);

development of methods and software for solution of problems of determination on the operating expenses of VTsKP and organization of mutual accounting with users (for automation of the work of the planning and dispatcher service of the VTsKP);

development of software for gathering and analysis of data on operation of the software and organizational aids of VTsKP (for automation of the work of the operating services of the VTsKP).

It was also noted that an imperative condition for successful solution of these problems is development of a unified standard information model of the VTsKP that contains data on the computer resources, users, tasks to be solved and so on. This model should rely on a unified system of standards for all types of documentation in the activity of the VTsKP, including a system of indicators of the operating efficiency of computer resources.

Among the most interesting reports devoted to development and operation of software for management of solution of applied problems, the following should primarily be named: "Automated System for Support of Processing Logically Related Jobs" (V. D. Gordiyuk and L. N. Olefir, Kiev), "An Tool Complex for Design and Realization of Data Processing on Computers" (V. I. Dvortsin and G. S. Prokudin, Kiev) and "Software for Support of Increased Data Processing Reliability in YeS Computers" (I. A. Kirillov, Kiev).

In discussion of these reports, the seminar participants noted the increasing timeliness of development and operation of a given type of software, which was determined by the development of distributed data processing systems based on multimachine complexes and computer networks. Recommendations on further improvement of these aids for development of a number of problem- and machine-oriented program complexes that permit management of real-time interaction of applications and systems processes in computer networks, were advanced to the developers of software for management of solution techniques. The latter is especially important for supporting effective joint operation of central computers with user stations (terminal complexes and intelligent terminals based on mini- and microcomputers).

A number of reports read at the seminar were devoted to problems of development of the hardware base of VTsKP and computer networks. Thus, the practically realized and operating radio channel computer network was reported in "Design Principles and Architecture of the DISKRET Computer Network Based on Packet Radio Communications" (A. M. Luchuk, R. G. Ofengenden and S. G. Bunin, Kiev). The attention of listeners was directed to the high technical-economic and operating characteristics of the hardware and software of the network (see UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No. 5, 1983, pp. 37-44 for more details).

The report "The PROTsESS Local Network: Experience in Realization and Operation" (S. S. Shalugin, Ya. P. Doriy and A. R. Shkolyarenko, Kiev) seems to be a continuation of a conversation on high-speed computer networks, but in regards to a new communication systems for development of local networks. The speakers talked about the experience of development and operation of hardware (network controllers, network interfaces and communication channels) that provide integration of different computers into a local network and that realize data exchange procedures between these computers over a high-speed channel (coaxial cable). The data transmission hardware in the PROTsESS network, developed and turned over for experimental operation, provides high transmission speed (up to 1 Mbit/s) and connection of different types of computers--YeS, BESM-6 and SM--to the network. The software, designed for management of network resources and dispatching of user jobs, is being debugged at the same time.

It is natural that in the course of discussion of the latter two papers numerous questions arose, the essence of which mainly reduce to the following: were the areas of efficient use of new hardware determined? What protocols were selected for organization of intercomputer exchange? How complicated is management of local network resources? How is interaction of applied processes organized? These and other questions indicate both the timeliness of development of new and promising intercomputer communication aids and the need for further concentration of efforts on investigation of the architecture of local network hardware and software in development of methods of selection and analysis of application of new data transmission and processing aids under specific conditions.

Problems of improving existing and development of new organizational aids and hardware for the computing process were considered at the seminar, mainly during discussion of the reports "Problems of Development of Organizational Support and Hardware and Organization of Operation of the Computer center" (A. D. Savchenko, Kiev) and "Experience in Organization of the Computing Process at VTsKP With Combination of Batch and Teleprocessing Modes" (A. M. Tovashev and V. V. Lukin, Moscow).

A common summary on these reports produced the opinion that different organizational aids are developed and introduced more successfully if there is a common strategy at the VTsKP in improving the organization of the computing process, which takes into account to the maximum extent the technical equipping, personnel support, makeup of users and also the functional characteristics of a specific VTsKP. Strict analysis that provides clear establishment of the priorities of goals for improving organizational aids for the computing process, evaluation of the consequences of introducing them and prospects for further development, is especially needed here.

Unfortunately, the administrators of VTsKP have not yet been provided with the correspoinding methods to the proper degree, which gives special importance to generalization of positive experience in support of VTsKP with organizational aids and hardware, development of document circulation systems and improvement of work with cadres.

Specific positive experience in solution of these problems has been accumulated at the Computer Center, Institute of Cybernetics imeni V. M. Glushkov, Ukrainian SSR Academy of Sciences, at the Tula VTsKP and at the Tomsk, Minsk, and Tallin VTsKP. The problem consists in familiarization of specialists of different computer centers with this experience and the fastest introduction of advanced methods of organization of the computing process. Of course, it is primarily necessary to eliminate the terminological disagreement of developers and to introduce unified concepts and standards and unified document circulation for different computer centers. In this regard the seminar participants approved the proposals of colleagues of the Tula VTsKP I. B. Viner, B. A. Kozinskiy and L. I. Mokhova on conducting a broad discussion and subsequent standardization of the terms in data processing techniques at VTsKP and computer networks (a list of terms and their interpretation is contained in the proceedings of the Fourth All-Union Conference on Collective-Use Computer Centers, held in September 1983 at Dushanbe).

A number of reports read at the seminar were devoted to problems of development and operation of instrument and technological software. Thus, the results of analysis of the operating characteristics of the best known interactive programming systems were presented and the ORAKUL system, realized and operated at a number of computer centers, which most fully satisfies contradictory requirements, was reported in the report "Interactive Programming Systems" (V. A. Kushnirov and F. A. Levchenko, Kiev) (thus, expansion of the command and service function systems makes it necessary to increase the capacity of the main memory, slows down the response of the system and so on).

The report "Access System to Data in Relational Data Base Management Systems" (R. P. Kramarenko and D. N. Kozlov, Kiev) was devoted to development of methods of investigation and analysis of the efficiency of data access systems in relational data bases. The authors suggested a version of a system that optimizes data retrieval and access operations.

The report "Development of Programming Systems in Machine Codes" (Yu. I. Mal'stev, Tomsk), in which a tool system that permits creation of a special operating environment in YeS computers when performing operations on connecting to YeS computers and debugging special apparatus, developed by the author, was reported, also aroused great interest of the seminar listeners. A number of effective, easily readjustable test programs for checking the operating reliability of peripheral devices and nonstandard equipment connected to YeS computers was developed on the basis of this system.

The experience of development and use of software tools, outlined in the mentioned and other reports, shows that this software should now be considered as an obligatory component rather than as service components of the software of the computer center. This conclusion confirms the active use of software tools in the determination of the optimum parameters of operating systems and applications program packages for analysis of the effectiveness of simulated and real computing processes, selection of optimum conditions and operating schemes of new hardware and software. A considerable saving of programmer labor and expenditures of machine time for development of new systems is achieved, the quality of design solutions is enhanced and deadlines for introduction of developments are reduced.

Summarizing the results of the seminar as a whole, one should dwell on a number of conclusions and recommendations formulated by its participants on further development of organizational aids, hardware and software for management of the computing process:

1. VTsKP must be considered as a technological system that includes hardware, software and organizational support in development of organizational aids, hardware and software for management of the computing process. These components should be interrelated and mutually balanced.

2. The vigorous development of aids for management of the computing process is a direct consequence of the complication of the hardware and software base of the computer center and of the increasing role of organizational support of data processing techniques. In other words, flexible organizational support that meets all requirements of creation and development of new data processing techniques

(high efficiency, low laboriousness, minimum executed operations and so on) should correspond to the constantly changing structure and capabilities of hardware and software.

3. Taking the dynamics of development of hardware and software and the degree of influence of this development on data processing techniques into account, one must constantly improve the software tools for development and analysis of planning and management decisions at VTsKP. The principal basis of all innovations at VTsKP should be a clearly defined strategy for its development.

4. Special attention should be devoted to predicting the development of data processing techniques as a whole and one should primarily consider the consequences of broad introduction of new communication aids and personal computers when determining the methods of achieving the goals of development of organizational aids, hardware and software for management of the computing process.

An exhibition of the prospectuses of new developments in management of the computing process, excursions to the Kiev Computer Center and creative contacts of the developers of aids with their users also contributed to the fruitful work of the seminar. It is gratifying that developments are exchanged during the seminar on agreements of scientific and technical cooperation, which considerably reduces the periods of introduction of developments into practice and permits more rapid determination of their weak and strong points.

The work of the seminar was conducted at a time when the broad scientific community noted the 60th birthday of the founder of Soviet cybernetics Academician Viktor Mikhaylovich Glushkov. The seminar participants visited the memorial room of Academician V. M. Glushkov at the Institute of Cybernetics that bears his name.

The next meeting of the seminar is planned in September 1984.

6521
CSO: 1863/122

INTERACTIVE MAN-MACHINE SYSTEMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 pp 125-126

[Article by V. I. Branovitskiy and N. A. Vlasenko]

[Text] The conference "Interactive Man-Machine Systems and the Efficiency of Their Application in the National Economy," organized by the Republic House of Economic and Scientific and Technical Propaganda (section 'Data Display Systems'), by the Institute of Cybernetics imeni V. M. Glushkov, Ukrainian SSR Academy of Sciences, by the Kiev Institute of Automation imeni XXV CPSU Congress and by the Kiev City Administration of NTO [Scientific Production Association] PRIBORPROM, was held in Kiev from 27 through 29 September 1983. Approximately 130 scientists and specialists from 28 cities of the Soviet Union participated in the work of the conference.

Approximately 60 reports and communications were heard and discussed at the plenary and sectional sessions. Five sections operated at the conference:

1. Problem-oriented interactive systems.

2. Automated teaching systems.

3. Information support of interactive systems.

4. Psychological problems of man-machine interaction.

5. Linguistic support of interactive systems.

Candidate of Technical Sciences V. I. Branovitskiy (Kiev), who substantiated the laws of the emergence of interactive systems that provide, on the one hand, the capability of active recruitment of final users to solution of their own problems and, on the other hand, automation of teaching the users of computers, opened the conference with the speech "Problems of the Creation of Man-Machine Dialog Systems" at the plenary session. A definition of dialogue was also given in the report, promising trends of development of interactive systems were determined and the main problems of developing man-machine interactive systems were formulated.

Five additional reports were also presented at the plenary session. The report of A. M. Dovgyallo (Kiev) "Experience of Design and Application of AOS [computer-aided instruction systems]," in which the work carried out at the Institute of

Cybernetics, Ukrainian SSR Academy of Sciences, jointly with the Scientific Research Institute of Problems of the Higher School, on development and use of AOS (introduced in more than 100 organizations throughout the country)--SPOK-VUS DOS YeS and OAS-VUZ OS YeS--was heard with great interest. The speaker also dwelled in detail on the problem of increasing the capabilities of interactive and teaching systems. The report of A. A. Molodtsov (Moscow) "SM Computers as Hardware for Interactive Systems" was devoted to description of the main characteristics of the family of SM computers, their software and the capabilities of using them in interactive systems.

The report of Yu. D. Broner (Moscow) "A Method of Estimating the Economic Effectiveness of Teaching and Interactive Systems" evoked great interest and widespread discussion. The method proposed by the speaker is based on the recommendations of the USSR GKNT [State Committee for Science and Technology], USSR Gosplan and the USSR Academy of Sciences and permits economic analysis of the operating expenses for selecting the direction of the most effective improvement of the structure of interactive systems. The method was approbated in the analysis of the economic effectiveness of the AOS EKSPO. The conference participants expressed the desire in their talks to confirm the proposed method and recommended that it be disseminated to interested organizations for review and subsequent confirmation by the USSR GKNT.

Reports devoted to problems of development of interactive systems based on the PROLOG-YeS system, evoked great interest at the sessions of section 1. It was noted in the report of S. P. Yefimenko (Kiev) "New Technique for Describing the Subject Areas of Interactive Systems (Based on the PROLOG-YeS Applications Program Package)" that the problem of developing the technique of simulating the subject areas is becoming especially acute in development of interactive systems; therefore, the corresponding presentation of knowledge for the subject area under consideration and the language of presentation of knowledge must be selected. It was suggested that PROLOG be used as this language. Problems of developing these systems were reflected in the report of N. A. Vlasenko "Problems of Development of Interactive Information Reference Systems Based on PROLOG Language." The report of A. R. Vilyums and P. S. Savel'yev (Riga) "Interactive System of Working With Program Packages Based on Axiomatic Representation of Knowledge," and also of Yu. D. Fedorov "Interactive System of Development and Tying Applications Program Packages to Specific Entities" were reported to problems of practical use of the PROLOG-YeS system, specifically, to development of interactive systems for working with program packages and tying applications program packages to specific entities.

A group of reports was devoted to practical developments. Among them can be the distinguished the report of L. S. Tyutikov (Perm) "Interactive Information Support System for Designers and Technicians," L. Ya. Bukharbayeva (Ufa) "Organization of Man-Machine Interaction at the Decision-Making Stage in Management of Experimental Production" and so on. The reports of G. M. Kot (Kiev) "Effectiveness of Task of Comparison to Prototype in Interactive Systems" and D. M. Rayko (Kiev) "Method of Designing Applications Interactive Programs in DSRV" and so on also aroused the interest of the participants.

The practical orientation of many developments reported in section 2, in the work of which a wide range of scientists and specialists participated, should

be noted. One can note the report of A. G. Chachko (Kiev) "Experience of Using Interactive Procedures in Teaching (on the Example of Training the Operating Personnel of Electric Power Plants)" in which the disadvantages of the simulation approach in development of simulators of electric power plant operators are shown on the example of the existing training center of the Tripolskaya GRES and the need to use a new method that takes into account the variety of control consoles of the GRES was substantiated.

The reports of L. P. Novitskiy (Riga) "The KONTAKT-OS Computer-Aided Teaching System" and of L. V. Zaytseva (Riga) "Automated Training Course in PL-1 Language and Method of Evaluating the Effectiveness of Its Use" aroused great interest. The KONTAKT-OS system is used for teaching and checking knowledge. The software permits connection of 16 displays. Teaching is now being carried out in 82 topics.

The attention of the conference participants was attracted by the talk of Yu. K. Sitnikov (Kazan), who shared the experience of work and development of automated instruction courses and integrated use of teaching hardware in his communication "The Structure and Characteristics of an Adaptive Teaching Course on Digital Computer Circuitry," and of V. I. Goncharov (Kiev), who talked in his report "Development of an Automated Interactive Simulator-Teaching System," on work and development of this system on the basis of SI language. A. P. Pisarenko (Kiev) "Development of Local TES and AES Simulators Based on Mini Computers," I. A. Grigorishin (Chernovtsy) "An RTK Teaching Course Based on SPOK," and R. M. Natslishvili (Kutaisi) "The Place of Dialog in Automation of Management and Teaching at Vuzes of Noncomputer Profile" presented interesting communications. The report of V. A. Yerokhin (Leningrad) "The Use of the Man-Computer Interactive System Based on the YeS 7920 Display Complex for Conducting Business Games in Teaching the Organizers of Aviation Plants Rational Actions in Failure Situations" must also be noted.

The reports of section 3 can be divided topically into two groups. The reports of V. N. Belov (Riga) "Axiomatic Models of Knowledge in Man-Machine Solution of Tasks," of A. A. Yeremy-Yeremenko (Kiev) "Methods of Describing the Subject Area of Data Base Design" and of N. M. Popova (Kiev) "Linguistic Problems of Constructing a Description of the Subject Area of a Data Base to be Designed," in which the principal methodological problems of design of models and languages for representation of knowledge were discussed, can be placed in the first group, devoted to representation of knowledge of systems to be designed. The second group, devoted to development of special interactive software, includes the reports of V. A. Temperanskiy (Kiev) "Interactive System for Hierarchical Type Data Banks" and V. O. Grechko (Kiev) "Joining Interactive Management Aids to Data Manipulation Hardware," in which the design principles and experience of using specific developments carried out at the Institute of Cybernetics, Ukrainian SSR Academy of Sciences, are described. These reports aroused great interest among the listeners, related to practical orientation of the proposed developments.

The work of section 4 was devoted to the use of the results of psychological and ergonomic investigations in development of interactive systems and to the problem of ergonomic analysis of their operation. The report of V. M.

Bondarovskaya (Kiev) "The Ergonomics of Man-Machine Interaction," in which the results of work on ergonomic design and analysis of video terminal systems were outlined, the basic directions of ergonomic development were determined and the results of practical realization of ergonomic designs were indicated, should be noted here. Experimental data were presented and analyzed in the report of Zh. Z. Levshinova (Kiev) "Ergonomic Analysis of the Functional Status of the Operator When Working With a Video Terminal."

The reports of N. I. Povyakel' (Kiev) "Cooperation of Specialists of Different Subject Areas in Development of Automated Interactive Systems," of L. N. Babanin (Moscow) "Psychological Problems of Automated Information Retrieval" and so on also evoked great interest.

Special attention was devoted in the reports presented in Section 5 to problems of automation of text entry, development of linguistic processors, development of interactive information retrieval systems and introduction of them in different spheres of the national economy and of development of interactive aids in machine translation systems.

Among the more interesting and informative reports of this section should be noted those of I. B. Shtern (Kiev) "Capabilities of Text and Thesaurus Interaction in Interactive Systems," of Ye. M. Kazakov (Moscow) "Interaction With the Data Base in Scientific Research and Developments," and of E. I. Korolev (Moscow) "Interaction in Terminological Data Bases (From Materials of Communication and Control Systems)." The report of P. P. Ivanov (Perm) "Linguistic Support of a Factographic Interactive System of Technological Designation" evoked special interest, since the given development is of important applied significance.

The basic directions of development of interactive systems were determined in the recommendations of the conference adopted at the concluding plenary session. It was noted that the new phase in the use of computer technology advances to the forefront the problem of presentation of diverse interactive aids to a wide range of final users whose professional interests are related to data processing on a computer. The need to increase the level of research in development of interactive systems based on methods and means of artificial intelligence and of expansion of the ergonomic development of video systems was emphasized.

The participants expressed a desire to hold these conferences regularly.

6521
CSO: 1863/122

SCHOOL-SEMINAR ON INTERACTIVE SYSTEMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83
pp 120-121

[Article by Aleksey Mikhaylovich Dovgyallo, doctor of engineering sciences, UkSSR Academy of Sciences Institute of Cybernetics (Kiev), and Mikhail Geogiyevich Khoshtaria, candidate of physical and mathematical sciences, GSSR Ministry of Finance Information and Computing Center (Tbilisi)]

[Text] The fifth, in succession, large meeting of specialists (2-10 April 1983, Kutaisi) on interactive systems development and applications was held in accordance with the general coordination plan of the Commission on Interactive Systems, headed in recent years by Professor I. I. Malashinin, director of the Information and Computing Center, USSR Academy of Sciences Institute of Atomic Energy. Just as before, the school was clearly all-union: 223 participants represented 93 leading scientific centers and organizations from 12 union republics. Interest in the school subject is indicated by the applications sent by more than 200 organizations to take part in its work. The Program Committee received about 500 papers. After thorough review and editing, 177 papers (plenary, main section and stand) were selected for the seminar program.

It is not possible to cite and describe each paper. So, we will briefly review the main directions which constituted the school subject and identify just the main informative groups of papers.

Three basic problems were discussed in the school: 1) evolution of theoretical aspects of man-machine interaction; 2) development of instrumental software and hardware facilities for supporting interaction; and 3) experience in development, application and introduction of various types of applied interactive systems.

Used more and more often as a theoretical apparatus in developing interactive systems are various applied linguistic models of a limited natural language: the apparatus of syntactic-semantic relations (Khar'kov school, group of papers on the DESTA system); algebra of finite predicates describing the relation between parts of word structure and that between word structure and the context surrounding it; the apparatus of junctive deep-syntactic representation in terms of Martem'yanov's verbal grammar and others. A considerable number of papers dealt with the frame approach to knowledge representation and

to implementation of machine understanding of a restricted natural language (Tallin and Tartu schools). In our view, promising developments are the theoretical apparatus which will allow representing scenarios of interaction as a system of tasks to be resolved jointly by the user and the computer (group of papers from the UkSSR Academy of Sciences Institute of Cybernetics), and the theory combining various pseudophysical logics and calculuses for cause-effect relations with the apparatus of describing frames or scenarios of dialog (USSR Academy of Sciences Computer Center, GeSSR Academy of Sciences Institute of Cybernetics).

Discussions on development of special instrumental software and hardware facilities for supporting interaction included:
--specialized software packages (PPP) intended for programming and maintaining a broad class of scenarios of interaction (family of packages SPOK, PPP YaOSD, DILYaPAS, DIADEL', STRATEG and others);
--facilities for maintaining speech interaction (paper by T. K. Vintsyuk, UkSSR Academy of Sciences Institute of Cybernetics), linguistic processors in the DESTA, TARLUS, ETAP and other systems;
--frame description languages (UTOPIST, F-language, FRL and KRL; and
--software packages for interactive computer graphics (an interesting family of these packages was presented in P. V. Vel'tmander's paper).

Facilities for general systems support of interaction in a system of virtual Unified System machines were discussed in a group of papers presented by the Scientific Research Institute of Computers (Minsk).

Compared to previous schools, there was more extensive representation in the fifth school (up to 30 percent of all papers) of the hardware and software basis for interaction on minicomputers and microcomputers. Promising in developing instrumental software and hardware facilities is the development of families of mobile evolving packages supporting the capability of gradual accumulation and circulation of practical interactive data bases and libraries of applications software.

Experience in developing, applying and introducing applied interactive systems for various classes of tasks was covered in half of all the papers and communications. In their review paper which aroused much interest among the seminar participants, Ye. L. Yushchenko and O. L. Perevozchikova generalized Ukrainian experience (as of the end of 1982) in developing and introducing interactive systems. It was based on processing 89 responses to a specially prepared questionnaire. In particular, they suggested distinguishing three stages of completeness of interactive systems: experimental application for needs of one's own organization, experimental operation in two-three other organizations and industrial operation over a number of years in several organizations. Judging by the papers presented in the school, interactive systems in experimental and industrial operation in the USSR implement:
--automated translation from one language to another;
--interaction with data bases in various kinds of ISS [information reference systems] and ASU [management information systems];
--computer-aided instruction;

--handling tasks in real-time systems (control of robots, manufacturing processes, flying vehicles and others);
--operation with simulation models;
--interactive processing of images and others.

A major trend in applied interactive systems development and implementation, in our view, is the increase in the number of efforts in which all three aspects indicated above are reflected: theory, instrumental facilities and industrial introduction (papers by Yu. D. Andresyan and coauthors, B. B. Timofeyev and coauthors, N. M. Veliashvili and coauthors, M. G. Pkhovelishvili and many others). In the same vein, efforts were performed and are underway on the DESTA natural language system, the subject of an expanded discussion (V. A. Lovitskiy and coauthors).

Along with the positive aspects mentioned above, a number of shortcomings were noted in a school resolution: lack of standards for interactive system programming languages, and the obvious shortages of training aids, journal publications and monographs covering interactive systems.

8545
CSO: 1863/80

PUBLICATIONS

NEW JOURNAL: MIKROPROTSESSORNYYE SREDSTVA I SISTEMY

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 p 17

[Text] Beginning in 1984, the USSR State Committee for Science and Technology and the All-Union Information Center on Equipment (VTsIO) will publish a new scientific and technical journal MIKROPROTSESSORNYYE SREDSTVA I SISTEMY.

The topics of the journal will include:

dissemination of Soviet experience in the use of microprocessors for automation of machines, equipment, instruments and production complexes;

publication of materials on the advances of CEMA countries in development and use of microprocessor equipment;

publication of problem-related articles, specifically, on problems of standardization, personnel training and also data on serially produced microprocessor equipment and new developments;

illumination of problems in development of software for microprocessor systems;

formulation of bibliographic surveys on the topics of the journal.

The frequency of the journal will be four issues per year. The volume of each issue will be 10 printer's pages. The price of a single issue will be 1 ruble 10 kopecks. The cost of an annual subscription will be 4 rubles 40 kopecks.

Subscriptions to the journal for 1984 will be taken through the Soyuzpechat' Department. Information about the journal has been presented in information brochure No. 2 attached to the catalog Newspapers and Journals for 1984. The index of the journal is 70588.

6521
CSO: 1863/122

TABLE OF CONTENTS: UPRAVLYAYUSHCHIYE SISTEMY I MASHINY NO 5, 1983

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5,
Sep-Oct 83 pp 1-2, 128

[Text] GENERAL PROBLEMS OF CONTROL SYSTEM STRUCTURE

SYSTEM HARDWARE AND AUXILIARY EQUIPMENT

8545
CSO: 1863/80

TABLE OF CONTENTS: UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, NO 1, JAN-FEB 84

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 84 pp 1-2, 128

[Text] COMPUTERS AND AUXILIARY EQUIPPING OF SYSTEMS

MICROPROCESSOR EQUIPMENT AND ITS APPLICATIONS

6521
CSO: 1863/122

PERSONALITIES

TRIBUTE TO GLUSHKOV

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 5, Sep-Oct 83 pp 122-123

[Article by Yuriy Georgiyevich Krivonos, candidate of physical and mathematical sciences, UkSSR Academy of Sciences Institute of Cybernetics (Kiev): "Tribute of Profound Respect"]

[Text] The country's scientific community widely celebrated the 60th anniversary of the birthday of Academician Viktor Mikhaylovich Glushkov, the prominent scientist and organizer of science, Hero of Socialist Labor and laureate of the Lenin and USSR and UkSSR State Prizes.

In accordance with the program developed, a number of measures devoted to this famous date were taken at the UkSSR Academy of Sciences Institute of Cybernetics imeni V. M. Glushkov and in Kiev. The first of these was the joint session of the Bureau of the Department of Mathematics and Cybernetics of the UkSSR Academy of Sciences, the Bureau of the Scientific Council on the "Cybernetics" Problem of the UkSSR Academy of Sciences and the Scientific Council of the Institute of Cybernetics. Academician V. S. Mikhalevich, director of the UkSSR Academy of Sciences Institute of Cybernetics, gave a report on Academician V. M. Glushkov's life and scientific and sociopolitical activity. Addresses were given to the attendees by Academician I. K. Pokhodnya, vice president of the UkSSR Academy of Sciences; academicians Yu. A. Mitropol'skiy and I. I. Lyashko; V. I. Tsyba, general director of the Kristall PO [Production Association]; and Kiev party and social organization representatives.

On V. M. Glushkov's birthday, 24 August, a memorial plaque was dedicated at house No. 15a on Yaroslavov Val Street, where the scientist lived and worked during the last 10 years of his life (from 1972 through 1982). Warm words about Viktor Mikhaylovich were spoken in both the address by D. B. Golovko, secretary of the Kiev City Committee of the Ukrainian Communist Party, who opened the meeting, and in the addresses by Academician V. I. Trefilov, vice president of the UkSSR Academy of Sciences; Academician A. A. Dorodnitsyn, member of the Bureau of the USSR Academy of Sciences Department of Mathematics and director of the USSR Academy of Sciences Computer Center; Academician V. S. Mikhalevich, UkSSR Academy of Sciences; and L. I. Litvinenko, member of the LKSMU [Ukrainian Komsomol] committee, UkSSR Academy of Sciences Institute of Cybernetics imeni V. M. Glushkov. Meeting attendees included S. I. Gurenko,

deputy chairman of the UkSSR Council of Ministers; K. M. Sytnik, vice president of the UkSSR Academy of Sciences; Col-Gen V. S. Rodin, chief of the Political Directorate, Red Banner Kiev Military District; and numerous representatives of the scientific community and workers in Kiev.

Earlier in the day, at the square in front of the main entrance to the UkSSR VDNKh [Exposition of National Economic Achievements], many people also attended the dedication of the memorial plaque on the building, opening the avenue named for Academician V. M. Glushkov.

On 6 September 1983, an extended session of the UkSSR Academy of Sciences presidium, dedicated to V. M. Glushkov's 60th birthday, was held. V. M. Glushkov's outstanding contribution to progress in basic research in mathematics, cybernetics and computer engineering was noted in addresses by Academician B. Ye. Paton, president of the UkSSR Academy of Sciences; Academician V. S. Mikhalevich; A. K. Romanov, deputy chairman of the USSR State Committee on Science and Technology; academicians V. S. Semenikhin and A. A. Samarskiy; Academician I. I. Lukinov, vice president of the UkSSR Academy of Sciences; and Academician V. K. Kabulov, UzSSR Academy of Sciences. The theory of digital automata and discrete transforms, developed by the scientist, just as the macro-pipeline organization of the computing process, suggested by him, laid the foundation for the general theory of computers and systems. A number of domestic models of computers and control systems, and new technologies for designing and programming them, which have been used extensively in the national economy, were developed under V. M. Glushkov's direction.

The UkSSR Academy of Sciences Presidium approved measures dedicated to the 60th anniversary of V. M. Glushkov's birthday and recommended the UkSSR Academy of Sciences Scientific Council on the "Cybernetics" Problem, the Bureau of the Department of Mathematics and Cybernetics of the UkSSR Academy of Sciences, and the UkSSR Academy of Sciences Institute of Cybernetics imeni V. M. Glushkov consider the question of publishing scientific instructional materials on the study of the creative legacy of the scientist and books of recollections about him, as well as the holding of Glushkov readings. It was recommended that the UkSSR Academy of Sciences scientific institutions make extensive use of the works by V. M. Glushkov in developing and introducing automated control and data processing systems as well as work engineering complexes.

In August-September of this year, a number of all-union and republic scientific conferences dedicated to the memory of V. M. Glushkov were held.

To commemorate the 60th anniversary of this prominent scientist's birthday, the USSR Ministry of Communications issued a colorful envelope with his portrait.

8545
CSO: 1863/80

- END -